DRIVER MANAGEMENT FOR LESS-THAN-TRUCKLOAD CARRIERS

A Thesis
Presented to
The Academic Faculty

by

Burak Karacık

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2007

DRIVER MANAGEMENT FOR LESS-THAN-TRUCKLOAD CARRIERS

Approved by:

Martin W. P. Savelsbergh, Co-advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Alan L. Erera, Co-advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Özlem Ergun
School of Industrial and Systems Engineering
Georgia Institute of Technology

Cynthia Leach
Enterprise Services, Network Development Group
YRC Worldwide

Chelsea C. White III
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: 20 December 2006

To Mom,

You mean the world to me...

&

To Dad,

I wish you were there to see this...

ACKNOWLEDGEMENTS

I was privileged to work with Alan L. Erera and Martin W. P. Savelsbergh. They supported me in every stage of graduate life, not only in good times, but especially in dire ones. I had precious conversations on academic matters with them, which taught me how to think about difficult problems, think about different facets of complex systems and to plan ahead. They made me realize the potential I have. I am grateful to have these incredible researchers and concerned people as my advisors.

I am grateful to the three financial contributors of this research: the less-than-truckload carrier that also provided the data for the computational experiments, Sloan Foundation Trucking Industry Program and Michael E. Tennenbaum.

Özlem Ergun was not only faculty, but a friend to me. She always had, even in her busiest times, a few minutes to encourage me to continue my path. I have learnt a lot from our insightful discussions, not only on academic topics but about life matters as well.

I am delighted Chelsea C. White III and Cynthia Leach accepted to be on my thesis committee. Their feedback and support was valuable to me.

I appreciate R. Gary Parker's assistance very much. He was a fatherly advisor, who solved every academic conflict I had and he was very helpful in finding financial support.

I am indebted to Michael Hewitt for helping me with the computational experiments towards the final stages of the thesis.

Over the four and half years at Georgia Tech, I have met many extraordinary people, whom I am proud to call as friends. I would be amiss if I did not mention a few, who have touched my soul: Ron Billings, Jennifer Brown, Serhan Duran, Murat Eröz, Yetkin İleri, Elçin Kentel, Gültekin Kuyzu, David Lewis, Okan Örsan Özener, Rebeca Sandino, and Dylan Shepardson. Their support, encouragement and friendship made Atlanta feel like home and life at Georgia Tech much more enjoyable.

There are two people in my life, without whom I would not be here. Their unconditional

love and support is the reason that has always kept me going. Mom, Dad, there are no words to express my gratitude. Thank you very much for everything you have done for me and I love you so much...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The trucking industry is vitally important to the economy. It provides an essential service by transporting goods from business to business and from business to consumer. The less-than-truckload (LTL) industry is an important segment, serving businesses that ship quantities ranging from 150 lbs to 10,000 lbs. Large LTL carriers use thousands of drivers daily to move loads between terminals in the linehaul network, and each driver may be used for multiple consecutive load dispatches between rest periods. Like small package and post carriers, LTL carriers are freight *consolidators*, combining and recombining shipments from many customers into truckloads of freight at various terminals. In this way, LTL carriers spread transportation costs (*e.g.*, vehicle and operator costs) over many customers.

Driver wages are a major component of transportation costs. Consequently, cost-effective driver management is of crucial importance for the profitability of LTL carriers. This thesis investigates a variety of issues related to driver management. The thesis has a strong focus on practice. That is, much of the research concentrates on developing methodologies and implementations that can, without too much effort, be converted to commercial strength decision support tools, and on conducting studies that generate and provide useful practical insights.

Chapters II and III describe a scheme for the dynamic scheduling of linehaul drivers developed for one of the largest LTL carriers in the United States. Dynamic driver scheduling is challenging because driver schedules must satisfy a complex set of rules, most of which are specified by government regulation to ensure highway safety. In addition, trucking moves, unlike commercial airline flights or train dispatches, are not pre-scheduled; typically, a truck is dispatched when a sufficient amount of freight has accumulated at a terminal and truck capacity can be utilized effectively. The technology developed in this dissertation combines greedy search with enumeration of time-feasible driver duties, and is capable of

generating cost-effective driver schedules covering 15,000–20,000 loads in a matter of minutes. The driver schedules satisfy a variety of real-life driver constraints, including U.S. DOT hours-of-service requirements (Chapter II) and union rules (Chapter III). Computational experiments justify the algorithmic design choices, and a comparison with real-world dispatch data from the LTL carrier indicates that the technology produces high-quality driver schedules.

Chapter IV describes a tactical tool for determining the allocation of drivers in a trucking terminal network. One of the key tactical questions faced by an LTL carrier is how many drivers to locate (or domicile) at each terminal. Determining an effective driver allocation can be especially difficult due to union rules. Most carriers are unionized and a portion of their drivers, called bid drivers, can only move loads between their domicile (home location) and a designated region. The driver allocation technology developed determines the number of drivers to allocate to each terminal as well as the designated region for each bid driver. Computational experiments using truck movement data representative of operations at a major U.S. LTL carrier demonstrate the effectiveness of our domiciling methodology, and show that union restrictions may result in substantially larger driver fleets, in some cases even up to 50% larger.

Chapter V, the final part of the thesis, investigates questions related to the number of drivers required to fulfill a given set of loaded truck moves in a more academic setting in order to obtain some fundamental insights. To facilitate the analysis, some simplifying assumptions are introduced: the terminal network consists of only two terminals and the exact dispatch times of the loaded moves are known. The goal is to determine the minimum number of drivers required to cover all the loaded moves, and the resultant dispatch schedule for these drivers. The problem is analyzed with a number of different restrictions imposed on driver schedules. Without any restrictions on schedules, the problem is shown to be polynomially solvable. It remains an open question if the problem remains polynomially solvable when drive time restrictions or duty time restrictions are imposed on the driver schedules. For these more restricted variants, several easily computable lower bounds on the minimum number of drivers are derived, integer programming formulations are presented,

and fast heuristics are designed and analyzed. A computational study provides insights into the quality of the lower bounds, the quality of the heuristics, and the computational requirements of the integer programming formulations.

CHAPTER I

INTRODUCTION

1.1 Less-than-Truckload Freight Transportation

The trucking industry is vitally important to the economy. It provides an essential service by transporting goods from business to business and from business to consumer. The less-than-truckload (LTL) industry is an important segment, serving businesses that ship quantities ranging from 150 lbs to 10,000 lbs, *i.e.*, less-than-truckload quantities. Large LTL carriers use thousands of drivers daily to move loads between terminals in the linehaul network, and each driver may be used for multiple consecutive load dispatches between rest periods.

Like small package and post carriers, LTL carriers are freight *consolidators*, combining and recombining shipments from many customers into truckloads of freight at various terminals. In this way, LTL carriers spread transportation costs (*e.g.*, vehicle and operator costs) over many customers.

Although it holds a smaller portion of the for-hire truck transportation market share than the truckload industry, LTL transportation is an important industry. In United States, LTL transportation generated $21 billion in revenues in 2001, which constitutes 22.34% of the total revenues generated by the major segments of the trucking industry [5]. As of 2006, LTL transportation is gaining more importance as major small package carriers FedEx and UPS started offering LTL service after acquiring medium-sized LTL carriers.

1.2 LTL Operations

Unlike truckload carriers, LTL trucking firms operate fixed terminal networks to provide service. Carriers pick up shipments from various shippers in a relatively small geographical area, say a city, and transport them to a terminal serving the area, referred to as a satellite or *end-of-line* terminal. These end-of-line terminals (EOLs) serve as sorting centers and

consolidation facilities for outbound freight. Since there usually is not enough freight collected at an EOL terminal to build full truckloads to EOLs serving other areas, another layer of consolidation is introduced in the system. Outbound freight from an EOL is sent to a terminal that consolidates freight from many EOLs, referred to as a *breakbulk* terminal.

A movement of freight between two connected terminals requiring a single driver will be referred to in this dissertation as a *load*, consistent with practice in industry. Breakbulk terminals handle enough freight to build and dispatch cost-efficient loads, *i.e.*, loads that completely, or almost completely, fill up two trailers. A typical shipment travels from an origin EOL to an origin breakbulk, then to a destination breakbulk and finally to a destination EOL. This hub-and-spoke network is referred to as the (main) *linehaul network*. When the distance between the origin and destination breakbulk of a load is too long for a single driver to serve while satisfying the driving hours allowed by regulation, a sequence of intermediate stops is introduced at so-called *relay terminals*. Usually, relay terminals are breakbulk terminals on the path from one breakbulk to another, although they may be special facilities. To ensure high service levels, the load is transferred at a relay terminal to another driver and continues with minimal delay. A large national LTL carrier in the United States might operate 300 to over 600 EOL terminals, with approximately one breakbulk for every 20 EOLs.

The path of an individual freight shipment through an LTL terminal network is identified by the sequence of terminals where the shipment is handled, where handling refers to unloading and loading of the shipment into some trailer. For freight shipments moving between each origin EOL-destination EOL pair, a so-called *load plan* specifies the path for each shipment. If enough freight is available, it may be possible at an origin breakbulk to build a load for a destination EOL terminal and bypass the destination breakbulk specified by the load plan. Loads dispatched from an origin breakbulk to a destination EOL terminal are called *direct loads*, because they do not need to be handled at an destination breakbulk. LTL carriers prefer that loads from breakbulks are dispatched directly to destination EOLs, since doing so reduces handling costs and time (which in turn increases the chance of delivering before the due date). Usually, a direct load follows the same physical path as a regular

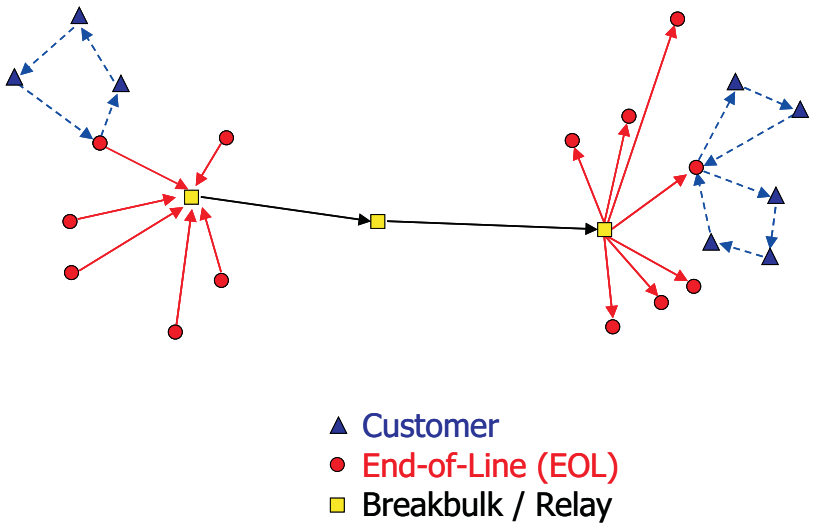

Figure 1.1: LTL network operations

load through the linehaul network in order to allow driver changes at relay points, but it is not handled until it reaches the destination satellite. Hence, direct loads may not decrease transportation costs. An important advantage of a direct load is that it typically reduces the time freight spends in the linehaul system (because it is handled less frequently), which improves service. Since the freight demand patterns observed by LTL carriers are fairly regular, LTL carriers usually identify and operate direct loads on certain origin-destination pairs. Direct loads from an origin EOL to a destination breakbulk are possible, but are used less frequently. Furthermore, if trailers are used for pickup and delivery of freight, then it may be possible to build loads destined directly for the origin breakbulk without requiring unloading and loading at an EOL terminal. This practice, which reduces handling cost and may improve service, is another reason why breakbulk-EOL direct loads are more likely to occur than EOL-breakbulk direct loads.

LTL carriers pack freight shipments into 28-foot trailers (or pups) or 48-foot vans. Typically, a single driver will pull either a single 48-foot van or two 28-foot trailers. One advantage of the use of 28-foot trailers is that they allow a single driver to pull 56 feet of trailers. In addition, the use of 28-foot trailers also improves service and reduces handling costs. Since it has less space, a 28-foot pup can be filled more quickly and thus may be

ready for dispatch sooner than a longer trailer. By combining trailers with different final destinations, it is therefore possible to build loads that can be dispatched earlier. Furthermore, filling an entire 28-foot trailer with shipments with a common final destination (which is more likely to occur with a 28-foot trailer than a 48-foot van) allows direct loading and simple drop-and-hook operations at the next terminal as opposed to loading and unloading. Effectively exploiting the advantages of the use of 28-foot trailers requires proper *pup matching* at breakbulks and relays, *i.e.*, deciding which 28-foot trailers to combine into loads.

Once a load is ready for dispatch (*i.e.*, all the shipments are loaded into the trailer(s) or van), at the so-called ready time, the shipment with the tightest service requirement implies a latest time that the trailer (or van) needs to be dispatched, called *cut time*, given the physical path that the load will follow through the linehaul network. A load can be dispatched any time between its ready time and cut time.

Due to imbalances in freight flows, it is impossible to avoid empty trailer repositioning moves in most LTL operations in practice. To a certain degree, long-run trailer balance at all terminals occurs naturally in LTL systems if drivers usually pull a van or two trailers, loaded or empty, whenever they are dispatched between terminals. In this case, since the driver assigned to any outbound load likely arrived inbound with trailers, trailer balance is generally preserved simply by ensuring that drivers are positioned correctly to move loads.

Although the load plan specifies how a carrier plans to move freight through its terminal network, a crucial component, one that has a significant impact on overall performance, is still missing: dispatching drivers with loaded or empty trailers through the linehaul network.

In LTL trucking operations, each driver has a home terminal, or *domicile*, where he is based, and to where he must return with some regularity. Typically, all breakbulk terminals serve as domiciles for drivers. A few satellite terminals that are located geographically distant from any breakbulk terminal or ones that have significant outbound flow may also have domiciled drivers as well.

Drivers must abide by a large number of rules that limit their activities. The first source of such rules is the U.S. Department of Transportation (DOT), which limits driving hours

and duty hours to ensure highway safety. After a driver reaches one of these limits, he is forced to *rest* (sleep) and stay off-duty for a minimum amount of time. There are other regulations imposed by the U.S. DOT that limit the number of hours that a driver can drive in 7 and 8 consecutive days. While these rules are important for long-term planning, they have less effect on shorter planning periods and will not be considered in this study. To maintain high levels of freight velocity through the linehaul network, it is clear that LTL carriers must actively coordinate driver rest periods so that loads that are ready to move are not waiting excessively because of resting drivers. Furthermore, driver rest periods may lead to direct and indirect operational costs for carriers. Each time a driver rests away from his domicile, the carrier incurs an extra cost (*foreign bed cost*). Since drivers typically prefer to rest at home, driver quality of life is increased and driver turnover costs are decreased by returning drivers to domiciles for rest frequently. For these reasons, LTL carriers typically prefer to ensure that no driver rests away from domicile more than twice in succession.

The second source of rules restricting driver schedules are contracts with labor unions. Most carriers are unionized and the labor unions have negotiated work rules that restrict, sometimes severely, the use of drivers at terminals. Unions require that a percentage of the LTL carrier's drivers are designated as *bid drivers*, whose dispatches are restricted. Each bid driver has a *bid terminal*. This terminal defines a region in the network to which the driver may be dispatched, and bid drivers are given priority on load dispatches to and from the bid terminal region. Bid rules and priorities bring regularity to the drivers' work schedule, since bid drivers only travel between their domiciles and the region defined by their bid terminal, and thus improve quality of life for drivers. However, since such rules restrict the dispatching decisions of the carrier, they lead to additional operational cost.

1.3 Decision Support for LTL carriers

This dissertation will focus on developing tactical and operational decision support for LTL carriers. In this section, we describe the most important tactical and operational decision problems faced by carriers, including the driver fleet sizing, allocation, and scheduling problems that will be considered in detail herein.

1.3.1 Service Network Design (Load Planning)

As mentioned in Section 1.2, the load plan specifies the route through the linehaul network that each freight shipment will nominally follow, as well as where consolidation occurs along the path. Construction of a cost-effective load plan is a tactical decision problem, and an example of a class of decision problems denoted *service network design*. Based on the cost of handling freight at breakbulk terminals and the freight flow between each origin-destination pair, the routes are determined and opportunities to build direct loads are discovered.

Most of the operations research literature focused on the LTL trucking industry and other consolidation transportation services has addressed this tactical freight flow planning. Crainic ([6] and [7]) provides excellent overviews of the large body of research in this area. Recent research considers time-critical transportation services such as those provided by the small package express industry (see *e.g.*, Barnhart and Schneur [3] and Armacost et al. [1]). Dynamic service network design has received little research attention so far, but is gaining in popularity and importance (see *e.g.*, Papadaki and Powell [13] and Crainic et al. [9]).

1.3.2 Driver Fleet Sizing and Allocation

Another set of major questions faced by LTL carriers is to determine at which terminals in the terminal network to domicile drivers, and the number of drivers to domicile at each terminal. Clearly, such decisions are intermeshed with operational driver scheduling decisions, and the rules (regulatory and union) that restrict these operational decisions. The primary complication of driver allocation is the fact that an individual driver may cover loads during the course of his dispatch schedule that are not outbound from or inbound to his domicile terminal. In such a case, this driver is helping drivers domiciled at another terminal serve their loads. Non-bid drivers, who do not have restrictions on the terminals to which they may be dispatched, may be able to serve a large variety of loaded dispatches from many non-domicile terminals. It is almost impossible to predict either the level or location of such interaction without a detailed simulation of operational dispatch decisions.

Therefore, simple allocation rules such as allocating drivers proportional to the outbound or inbound freight volume may be approaches that are too simplistic.

Although there are several papers in the operations research literature focused on resource allocation in the LTL trucking industry, most of them focus on equipment (*i.e.*, tractors and trailers) rather than drivers. The research of Cranic and Roy [8], Powell [14] and Çalışkan and Hall [4] considers drivers and focus on issues such as returning drivers home. However, these references ignore the fact that drivers are restricted by U.S. DOT restrictions and union rules altogether.

In Chapter IV, we develop a driver fleet sizing and allocation approach based on an algorithm that allocates drivers to domiciles using an iterative scheme. We address three tactical questions related to the allocation of drivers in a terminal network: how many drivers should be located at each terminal, what percentage of these drivers should be bid drivers, and what should be the bid terminal of each bid driver? The iterative algorithmic scheme uses a driver scheduling heuristic each iteration to determine a near-optimal way to use a given allocated fleet, such that each driver dispatch schedule satisfies U.S. DOT restrictions and union rules.

1.3.3 Driver Scheduling

Driver scheduling is a multi-dimensional large-scale operational problem that each LTL carrier faces. Effectively managing driver schedules is one of the most challenging aspects of an LTL operations and there are various complicating factors.

The first source of complexity is the large number of rules drivers must abide by. As described earlier, operational driver scheduling is constrained both by U.S. DOT regulations on hours-of-service and by union rules regarding usage of bid drivers. Clearly, any realistic driver scheduling scheme should consider these rules and restrictions. Where drivers rest and how often they return to their domiciles are also important considerations, even when such rest decisions are not explicitly dictated by rules.

Another aspect of LTL transportation that complicates driver scheduling is the flexible dispatch time of the loads. The flexibility of dispatching any given load between its ready

and cut times adds to the problem complexity, since the problem becomes one of making the joint decision of which driver to dispatch with a load and when to dispatch a load. Driver scheduling is further complicated by empty repositioning decisions. When and where to dispatch drivers with empty equipment must be integrated into the determination of a loaded dispatch schedule.

Today, most LTL driver scheduling is performed locally by terminal managers who dynamically determine assignments of drivers to loads and dispatch times. Although driver scheduling is of primary importance for efficient and effective LTL operations, it has received relatively little attention by the research community. This is partly due to a historical tendency to apply operations research methodology to planning problems rather than operational problems, and partly due to the challenges of centralized driver scheduling at large LTL carriers, which at first glance seem to be almost insurmountable. Given the current low cost of computing power and the growing trend in the academic community to focus on dynamic operational problems, the time seems right to see what can be done to assist national LTL carriers with driver scheduling. In Chapters II and 3, we will present driver scheduling technology that is capable of determining effective driver dispatch schedules covering more than 17,000 loads with 3,500 drivers within a matter of minutes.

1.4 Literature Related to Driver Scheduling

In the area of transportation operator scheduling, there is a significant body of research literature focusing on crew scheduling and rostering problems for transportation systems operating fixed schedules, such as passenger airlines, transit systems, and passenger rail services. Again, most research addresses tactical planning problems and applies a set covering or partitioning integer programming model to choose a subset of partial schedules, solved exactly or heuristically by enumeration or column generation. Barnhart et al. [2] provides a thorough overview of airline crew scheduling; recent advances in this field have addressed tactical crew planning under uncertainty (see *e.g.*, Schaefer et al. [18]). Importantly, the solution times required by this general approach makes it difficult to apply in an operational setting with dynamically-changing data. Advances in solution speed using

specialized solution techniques still lead to long computation times. Elhallaoui et al. [10] reports computation times greater than an hour for scheduling problems with more than 1,500 tasks.

Perhaps the most closely related research to the driver scheduling technology proposed in Chapters II and III is the research focusing on dynamic resource/asset allocation problems, typically in the context of managing drivers for truckload transportation firms. The body of research by Powell and co-authors focuses primarily on methods for handling data stochasticity; see [15] and [17] for summaries of the most recent research. The adaptive dynamic programming approach applied to problems with a discretized time dimension outlined in papers by Godfrey and Powell [11, 12] appears to be the most promising approach for solving these problems.

Yang et al. [19] consider a dynamic truckload assignment problem and show, using a simulation, that myopic rolling horizon reoptimization policies can perform quite well when compared to an *a posteriori* optimization. Their work, however, ignores the resource time constraints crucial for driver scheduling decision-making. It should also be noted that truckload driver management problems are quite different from the LTL problem, since load requests do not require movement between distinct terminals and tend to arrive with less predictability than LTL loads.

Powell et al. [16] consider a deterministic LTL driver scheduling problem quite similar to the operational problem that we study in Chapters II and 3. After concluding that integer-programming techniques are impossible to apply due to problem scale, the authors instead apply an approximate dynamic programming methodology similar to those developed for stochastic problems. While the approach yields promising results with computation times in the range of an hour when all constraints are included, it relies on a discrete representation of time (with computation times that depend on the fineness of the discretization) and a discrete representation of the state of a driver, including the remaining drive hours. In our work, we will not rely on such discretizations of the problem.

1.5 Thesis Preview and Contributions

The remainder of this thesis is organized as follows: In Chapter II, we propose an operational driver scheduling scheme for LTL carriers. In Chapter III, we enhance the scheme by incorporating bids and union rules explicitly within the scheduling technology. A tactical tool that determines how to allocate a driver fleet to a terminal network is presented in Chapter IV. Finally, in Chapter V, we focus on developing a fundamental understanding of problems of driver fleet sizing, allocation, and schedule management given different variations of hours-of-service restrictions. To do so, we focus on a simplified LTL network consisting of two terminals and investigate and analyze both optimal approaches and suboptimal heuristics for the problem of minimizing the number of drivers required to serve a set of loads during a fixed planning horizon.

The main contributions of this thesis can be summarized as follows:

- We have developed a real-time operational decision support tool for scheduling drivers of less-than-truckload carriers. The main features of the tool are:
  - Makes dispatch decisions for more than 17,000 loads and more than 3,500 drivers in minutes,
  - Focuses on the number of drivers dispatched as well as driver utilization,
  - Incorporates empty repositioning decisions, and
  - Handles a representative set of union rules as well as important U.S. DOT regulations.

  Computational experiments demonstrate that the tool produces high quality solutions that compare favorably to solutions used by a major U.S. LTL carrier.

- We have developed a method for locating drivers in a terminal network. The method decides
  - The number of drivers to locate at a domicile, and
  - For the bid drivers located at a domicile, their bid terminal.

Computational experiments demonstrate that the method outperforms schemes based on allocating drivers proportional to the aggregate volume of freight from each domicile.

- We study the impact of union rules on the number of drivers required to serve the loads in the linehaul network and show that this impact can be significant, in some cases even leading to an increase of 50% in the number of drivers required.

- We analyze in an abstract setting the computational complexity of one of the fundamental problems in driver management: minimizing the number of drivers required to cover a given set of loads. The results include:

  - When there are no restrictions on driver schedules, the problem of minimizing the number of drivers is polynomially solvable.
  - When there are restrictions on driver schedules, fast and effective optimization-based heuristics can be developed.
  - When there are restrictions on drive time, an approximation algorithm with a performance guarantee of 2 can be developed, *i.e.*, the algorithm will always produce a solution with a value that is no more than twice the optimal value.
  - Integer programming models can be formulated for the variants with restrictions on the drive time and/or duty time.

CHAPTER II

DRIVER SCHEDULING

2.1 Introduction

In this chapter, we describe a dynamic scheme for the scheduling of linehaul drivers that we have developed for a large U.S. LTL carrier. The technology is capable of generating cost effective driver schedules satisfying a variety of real-life constraints in a matter of minutes. Large-scale dynamic scheduling problems with renewable resources like drivers are among the most challenging scheduling problems. A renewable resource is one that is available for use for a limited duration after some start time, but can be renewed by recharging (in this case, resting) for a minimum recharge duration. Virtually all operator scheduling problems faced by transportation service providers are complicated by renewable resources. However, unlike passenger airlines and train services, LTL driver scheduling is further complicated by the fact that LTL trucking moves are not pre-scheduled. The driving tasks to be assigned to drivers have flexible dispatch times, but some tasks are linked by precedence relationships. In addition, LTL trucking service is usually "demand-responsive"; if enough freight has accumulated at a terminal bound for a specific next terminal, a load is ready to be dispatched as soon as an appropriate driver can be assigned.

The goal of the dynamic driver scheduling scheme developed herein is to determine the best load dispatch times and driver-to-load assignments for some fixed-duration planning period (usually 48–72 hours, a time frame that we will justify later). The methodology is to be re-run periodically to take advantage of the availability of new data to improve existing plans. Importantly, the notion of the best solution is tricky to define in this context. The goal of the firm is to minimize the driver costs required to meet customer service targets, but the true costs of a driver schedule are often difficult to quantify.

The scheme we develop combines greedy search ideas with partial enumeration to determine cost-effective solutions in minutes. The scale of the problems typically encountered for

large trucking firms—a few thousand driver resources and between ten and twenty thousand dispatch tasks—renders a mathematical programming approach impractical; a column generation methodology, even one that leads only to a heuristic solution, would likely require too much computation time. Further, a non-greedy local improvement strategy such as tabu search is also difficult to implement successfully in this context, since small changes to a small set of driver schedules can propagate throughout the solution. Since we rely on greedy search, the primary research question we would like to answer is: can a creative application of greedy search be used to determine high-quality solutions for dynamic LTL driver scheduling?

The primary contributions of this chapter can be summarized as follows:

- This chapter introduces the *driver scheduling and load dispatching problem* (DSLDP), a generalization of the operator scheduling problems with fixed task dispatch times typically considered in transportation research and an important dynamic resource scheduling problem. The DSLDP is applicable not only to LTL carriers, but also to small package express carriers.

- This chapter describes a fast, effective heuristic combining greedy search with enumeration to determine cost-effective solutions to the DSLDP in minutes.

- This chapter provides a computational study that shows that (1) combining enumeration with greedy search can lead to significant improvements in solution quality over simpler greedy procedures for large-scale resource scheduling problems, and that (2) greedy approaches can be used in a dynamic framework to develop good solutions to the DSLDP.

The remainder of this chapter is organized as follows: Section 2.2 provides a high-level description of the driver scheduling challenges faced by LTL carriers, and Section 2.3 provides a formal problem definition for the DSLDP. Section 2.4 describes the developed solution approach, and Section 2.5 discusses issues in implementing the approach for dynamic decision-making. Finally, Section 2.6 presents the results of a computational study using the approach with historical data from a large U.S. LTL carrier.

2.2 Managing Driver Schedules

The discussion of LTL operations in Section 1.2 shows that the primary transportation tasks faced by carriers is the movement of combinations of one or more pups and vans, containing shipments with a variety of origins and destinations, between terminals that are *connected* in the linehaul network. Two terminals are said to be connected if the travel time between them is no greater than the maximum drive time allowed by the DOT. A movement of freight between two connected terminals requiring a single driver will be referred to as a *load*; as described in Section 1.2, a load will usually consist of a single 48-foot van or two 28-foot trailers. Each load has a starting terminal (origin), an ending terminal (destination), a ready time (the earliest time the load can be dispatched), and a cut time (the latest time the load can be dispatched). The ready and cut time of a load are determined by the shipments contained in the load. Before a load can be dispatched, all contained shipments must have arrived at the origin terminal and undergone any necessary processing such as unloading, sorting, and reloading. The latest time that a load can be dispatched depends on the shipment contained in the load with the most constraining service requirement, where a service requirement typically specifies the due date of the shipment at its final destination.

Loads are said to be *built* at the origin terminal. Building a load may involve physically loading shipments into a van or into one or more trailers, or may involve matching of pups, or may simply involve relaying a van or a combination of trailers. Since shipments are usually contained in multiple loads along the path from their origin satellites to their destination satellites, precedence relations exist between loads. A precedence relation exists between every pair of loads that contain a common shipment. Fortunately, due to transitivity of precedence relations, it suffices to track only precedence relations between "consecutive" loads.

Each load requires a driver to move it through the linehaul network. As mentioned in Section 1.2, each driver has a *domicile*, a home location where he must return with some regularity. If we consider a single epoch in time, each linehaul driver has a current *state* including a "next" location where he will become available for assignment, a ready-to-assign time at the next location, and remaining available drive and duty hours at the next location.

Drivers can only be used for a limited duration before they need to be renewed by taking a *long rest*, during which time the driver sleeps. The minimum long rest time is specified by DOT regulations. If a driver is currently resting, then the next location is his current location, his ready time is his *off-rest time*, and his remaining drive and duty hours are given by the maximum DOT limits allowed between long rests. If a driver is *en-route*, then the next location is the terminal he reaches when completing his current move, his ready time is the time at which he completes his current move, and the remaining drive and duty time are set appropriately.

Driver scheduling technology is designed to take load and driver information for some time horizon beginning at some time epoch and attempt to make the following related decisions:

- Which driver to assign to each load?
- When to dispatch each load?

Any driver scheduling system must consider a mixture of goals when assigning drivers to loads and deciding dispatch times, *e.g.*,

- Maximize the number of loads that are dispatched on time
- Minimize the number of driver long rests at non-domicile terminals (*foreign beds*)
- Construct driver *duties* (sequences of dispatches between long rests) with drive times close to the allowed maximum
- Construct driver duties with little or no waiting time between consecutive load dispatches
- Minimize the use of empty repositioning

Each of these goals relates to the fundamental trade-off faced by LTL carriers: service versus cost. Load dispatch windows ensure a desired level of service for each shipment. On the other hand, all driver-related goals tie back to costs, sometimes indirectly through

"quality-of-job" considerations that reduce costs related to driver turnover. For example, sometimes the frequency with which a driver must return to his domicile for rest is specified by contract. However, even when it is not, it is desirable for the carrier to return a driver to his domicile frequently to improve quality-of-life. Note also that minimizing empty repositioning is an explicit goal. Some repositioning is necessary to allow each load to meet its desired service level, but carriers know that it is extremely difficult, if not impossible, to determine the appropriate amount of empty travel. Therefore, they aim to at least control the empty repositioning introduced into the system.

In the next section, we present a more formal definition of a driver scheduling problem, which forms the basis for the algorithmic approach we have developed.

2.3 Driver Scheduling and Load Dispatching Problem (DSLDP)

The *driver scheduling and load dispatching problem* (DSLDP) is to determine a set of driver movement schedules for a set of loads during a given planning period. Each load is characterized by an origin terminal, a destination terminal, a ready time, a cut time, a set of preceding loads which must arrive at the origin terminal before a dispatch can take place, and a set of succeeding loads which can depart from the destination terminal only after the load has arrived there. A set of drivers is available to move the loads. Each driver has a domicile, a current location at the beginning of the planning period, and a ready time at the current location. A driver schedule consists of a sequence of moves (loaded or empty) with associated dispatch times and long rest periods with start times. Sequences of moves between long rests are known as *duties*; when a driver has been assigned to a duty, we refer to the duty as a *trip*. A feasible driver schedule must satisfy regulations governing maximum drive hours and duty hours between long rest periods. The objective of the DSLDP is to find a set of driver schedules with minimum total operating costs, such that all loads are dispatched between their ready and cut times.

Formally, we use the following notation to model the driver scheduling problem:

$\mathcal{T}$ The set of all terminals.

$\mathcal{C}$ The set of all connections.

$\mathcal{L}$ The set of all loads.

$\mathcal{D}$ The set of all drivers.

The following parameters are used to model DOT driver regulations:

τ_V The maximum drive hours between long rests.

τ_U The maximum duty hours (including waiting) between long rests.

τ_R The minimum duration (hours) of a long rest.

From this point on, we use current U.S. DOT driver regulations: $\tau_V = 11$, $\tau_U = 14$, and $\tau_R = 10$.

Each load $\ell \in \mathcal{L}$ has the following attributes:

ORIG(ℓ) The origin terminal.

DEST(ℓ) The destination terminal.

READY(ℓ) Earliest dispatch time.

CUT(ℓ) Latest dispatch time.

PREC(ℓ) The set of immediate predecessor loads.

SUC(ℓ) The set of immediate successor loads.

Note that an immediate predecessor load $p \in \text{PREC}(\ell)$ must have $\text{DEST}(p) = \text{ORIG}(\ell)$, and an immediate successor load $s \in \text{SUC}(\ell)$ must have $\text{ORIG}(s) = \text{DEST}(\ell)$.

Each driver $d \in \mathcal{D}$ has the following attributes, representing his state at the beginning of the planning period:

DOMICILE(d) The domicile.

LOCATION(d) The current location.

READY(d) Earliest dispatch time at the current location.

MAXDRIVE(d) Remaining drive time.

MAXDUTY(d) Remaining duty time.

Note that driver d may be enroute to the current location LOCATION(d) at the beginning

of the planning period. In this case, and in the case where a driver is waiting (not resting) at the current location, MAXDRIVE(d) and MAXDUTY(d) may be less than the DOT maximums. In the case in which the driver is resting at the beginning of the planning period, MAXDRIVE(d) $= \tau_V$ and MAXDUTY(d) $= \tau_U$.

Each connection in $\mathcal{C}$ links two terminals $a, b \in \mathcal{T}$ and has the following attributes:

- COST(a, b) The cost (in dollars) of driving (loaded or empty) between terminals a and b.
- TIME(a, b) The drive time (in hours) between terminals a and b.
- RTIME(a, b) The run time (in hours) between terminals a and b.

The set of connections defines the linehaul network. As mentioned earlier, if two terminals are connected, the drive time between them is no greater than τ_V, and there is frequent transportation between the two terminals. Depending on the drive time between two terminals, a driver may need short rests along the way. These short breaks are incorporated in the run time and count towards duty time. Note that RTIME(a, b) $\leq \tau_U$ for all connected terminals.

In addition to the cost incurred while driving between two terminals, a fixed cost is incurred for a driver resting at a location other than his domicile (foreign bed cost).

2.4 Solving DSLDP

Real-world instances of the DSLDP for LTL carriers are likely to be quite large, since the problem is to create driver schedules for the entire linehaul network based on the latest available information (status of all drivers, and all actual and forecasted loads). To ensure that driver schedules are not overly influenced by end-of-horizon effects, real-world instances should also plan for multiple days. Experience with historical dispatch data from a large U.S. LTL carrier indicates that typical instances may include 15,000–20,000 loads for dispatch, and thousands of drivers. Because of the scale of the problem, decomposing the problem into smaller parts and solving the subproblems with optimal methods is a potential approach to solving DSLDP. However, there is no natural geographical decomposition, since both the drivers and the loads span the entire U.S. and may cross the boundaries of any smaller

region defined for the subproblems. Therefore, to be able to construct driver schedules in an acceptable amount of time we have chosen to develop a solution approach that combines greedy search with enumeration.

An important feature of the solution approach is that we assign a complete duty to a driver each time we make an assign-and-dispatch decision; a simpler alternative would be to simply assign a single loaded move to a driver at a time. Assigning complete duties facilitates the evaluation of decisions since many cost-minimizing "goals" relate to complete duties. For example, a very "good" duty has at least 9 hours of driving, less than 2 hours of waiting time between moves, and returns the driver to his domicile. A second important feature of the approach is that we attempt to enumerate potentially large sets of prospective duties each iteration. However, unlike exact approaches for transportation scheduling problems that enumerate all partial solutions, we restrict the enumeration at each iteration to only those feasible duties that include a specific selected load as the initial loaded move. Further, instead of combining a subset of the partial solutions into a complete solution using a set covering or partitioning integer program, we simply make a greedy choice each iteration using a hierarchical scoring mechanism.

A high level description of the heuristic algorithm for the driver scheduling and load dispatch problem can be found in Algorithm 2.1.

Algorithm 2.1 High level overview of the heuristic algorithm.

while There exist undispatched loads **do**
  Select a load to be dispatched next.
  Generate a set of duties that include the selected load as the first loaded move of the trip
  Generate a set of trips by matching drivers with duties
  Evaluate the quality of each trip with respect to objectives
  Select and dispatch a high-quality trip
  Update system information
end while

For efficiency reasons, the generation of driver trips is split into two phases. We observed that generating a set of trips separately for each driver that includes the selected load as the first loaded move sometimes resulted in tremendous duplication of effort. In many situations, a number of drivers with identical characteristics (*e.g.*, ready time, remaining

driving hours, and remaining duty hours) is available at the origin terminal of the selected load. For each of these drivers, the same set of duties was generated through enumeration. By first generating duty sequences of loads and empty moves, and then matching drivers with duties, *i.e.*, checking whether the driver ready time, remaining driving hours, and remaining duty hours are compatible with the duty, a significant speedup was obtained.

Next, we discuss the key steps in the algorithm in more detail.

Load selection

We iterate over the loads in order of increasing cut times. That is, we always select the load with the earliest cut time (breaking ties arbitrarily) and attempt to dispatch it. This choice is motivated by the importance of meeting service. If a load is not dispatched before its cut time, at least one shipment in the load will not be delivered on time.

Duty generation

Once a load has been selected, we attempt to generate all time-feasible duties that include the selected load as the first loaded move. A time-feasible duty consists of one or more loads, each dispatched between its ready and cut time, such that maximum limits on drive time and duty time are not violated.

We generate time-feasible duties using a depth-first recursive procedure. For a load $\ell \in \mathcal{L}$, we construct all duties starting with ℓ. To control the number of trips generated, we restrict the waiting time between consecutive loaded movements to be smaller than a predefined value W. Sometimes, we will also truncate the recursion to limit the number of duties generated. Each recursive step starts with a *base duty*, initially just the load ℓ. Then, to enforce the limit on the waiting time, we find the loads originating at the last terminal in the base duty with a ready time that is less than the arrival time at this terminal plus W. For all such loads $\hat{\ell}$, we check if the duty consisting of the base duty and the load $\hat{\ell}$ is time-feasible. If so, we add the duty to the list of generated time-feasible duties and invoke a new recursion where the base duty is the newly generated duty, *i.e.*, load $\hat{\ell}$ is the last load in the extended base duty.

For a duty to be time-feasible, it must not violate the load dispatch windows, and the

drive hours and duty period length must be no greater than the DOT maximums τ_V and τ_U. Because of the dispatch windows, the duty may include waiting time. The waiting time of a duty depends on the dispatch times of each of the loads in the duty. Therefore, these dispatch times should be chosen in such a way that the total waiting time in the duty is minimum. Note that the set of dispatch times resulting in the minimum total waiting in a duty need not be unique. We set the dispatch time of the first load in the duty to be the earliest time which results in the minimum total waiting time of the duty, and the dispatch times of the remaining loads in the duty to the earliest feasible dispatch time. During the construction of time-feasible duties we monitor and update, if necessary, the dispatch times on the partial duty to maintain this (invariant) property.

Given a duty with minimum wait time, there still may be flexibility in the dispatch time of the first load in the duty, in the sense that departing later may not affect the total waiting time of the duty. We define the difference between the latest and the earliest dispatch times of the first load in the duty D that results in a duty with minimum total wait as the flexibility of the duty, denoted by ϕ_D. Figure 2.1 illustrates the concept of flexibility for a duty with three loads, A, B, and C. The bracketed ranges represent the feasible dispatch time windows between READY(l) and CUT(l) for each load l.

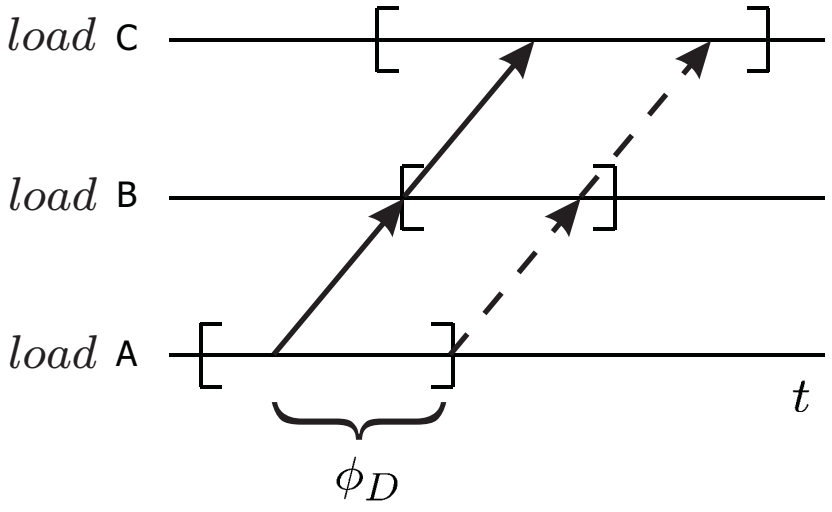

Figure 2.1: Flexibility ϕ_D for a duty with three loads; delaying the dispatch of load A no more than ϕ_D results in a minimum wait duty.

While it is trivial to determine whether a base duty D can be feasibly extended with load $\hat{\ell}$ considering only the drive hour maximum τ_V and the latest dispatch time $\text{CUT}(\hat{\ell})$, it is a bit more difficult to determine whether the duty hour maximum τ_U is not violated. Let σ_D denote the dispatch time of the first load of some base duty D, and let ω_D denote the duration of D. Further, suppose $\hat{\ell}$ is a candidate load for extending D which is feasible with respect to maximum drive hours and latest dispatch time. Let $t_{\hat{\ell}}$ be the run time of $\hat{\ell}$, and let $\delta_{\hat{\ell}}$ denote the width of its dispatch window: $\text{CUT}(\hat{\ell}) - \text{READY}(\hat{\ell})$. The duration of the duty created when $\hat{\ell}$ is added to D is computed as follows. First, we compute the beginning and end of the dispatch window of $\hat{\ell}$ relative to σ_D, denoted t_1 and t_2 respectively: $t_2 = \text{CUT}(\hat{\ell}) - \sigma_D$ and $t_1 = t_2 - \delta_{\hat{\ell}}$. Then, the dispatch time of $\hat{\ell}$ relative to σ_D, $h_{\hat{\ell}}$, is equal to $\max\{\omega_D, t_1\}$. If t_1 is strictly greater than ω_D, then there may *potentially* be a waiting time at the origin of $\hat{\ell}$. Let $w_{\hat{\ell}} = t_1 - \omega_D$ denote this potential waiting time. Some or all of this waiting time, however, may be eliminated by departing later at the origin of the first load of D. The dispatch time of the first load of the base duty can be delayed by at most ϕ_D, the flexibility of the base duty. Thus, the minimum waiting time at the origin of load $\hat{\ell}$ is $\bar{w}_{\hat{\ell}} = \max\{w_{\hat{\ell}} - \phi_D, 0\}$. Note that by departing later, the total wait time of D does not change, and that the total wait time of the extended duty is equal to the wait time of D added to $\bar{w}_{\hat{\ell}}$. We now have all the information we need to compute the duration of the extended duty. If the extended duty is feasible with respect to τ_U, we also compute its dispatch time and flexibility. Letting D_{ext} denote the extended duty, we have

$$\begin{aligned}
\omega_{D_{ext}} &= \omega_D + \bar{w}_{\hat{\ell}} + t_{\hat{\ell}} \\
\sigma_{D_{ext}} &= \sigma_D + (w_{\hat{\ell}} - \bar{w}_{\hat{\ell}}) \\
\phi_{D_{ext}} &= \min\{\phi_D - (w_{\hat{\ell}} - \bar{w}_{\hat{\ell}}), t_2 - h_{\hat{\ell}}\}.
\end{aligned}$$

Trip generation

Once all time-feasible duties are generated, we generate the set of all feasible trips by matching drivers available at the origin terminal of the selected load (and thus the origin terminal of all the duties) with duties, *i.e.*, by checking whether the driver ready time, remaining driving hours, and remaining duty hours are compatible with the duty. Note

that driver x has a compatible ready time with duty D if $\text{READY}(x) \leq \sigma_D + \phi_D$.

If no trips are generated for drivers at the origin terminal of the selected load, trips are created for drivers that start at another terminal and thus "bring in" a driver with an empty move. To control empty repositioning, this process is conducted in two phases. First, all drivers at a predefined set of *nearby* terminals are considered. Second, if no trips can be generated for drivers at nearby terminals, trips are generated for drivers at more distant terminals with historical empty repositioning to this terminal; this process considers terminals one at a time in order of increasing distance from the origin terminal.

Since one of the objectives is to minimize the number of foreign beds, additional trips are generated that add an empty move as the final leg to return a driver to his domicile when such a move is time-feasible. We note that we do not discard the original trip. These trips compete with all other generated trips.

Trip scoring and selection

Each generated trip is evaluated based on how well it contributes to the various problem objectives. Once the best trip is identified, it is scheduled for dispatch, *i.e.*, the driver and duty are dispatched to cover the initial load (and all other loads included in the duty).

Since the cost of a driver schedule is driven by many competing factors, defining an appropriate trip scoring and selection scheme is a challenge. After extensive experimentation with various schemes and weights for each objective, the following two-level hierarchical scheme was adopted. The hierarchical scheme reflects a difference in importance between the various goals. The primary performance indicators for a cost-effective operation plan are low empty repositioning miles and high driver utilization. A secondary indicator is the number of driver foreign beds.

Suppose trip T is formed by matching duty D to driver d. Let ν_T^l denote the loaded drive time of trip T and r_T^l the loaded run time of trip T. Further, recall that ω_D is the duration of trip T. Furthermore, let $\nu_T^{\max} \leq \tau_V$ be the maximum number of available driving hours for driver d; we note that $\nu_d^{\max} = \tau_V$ in the typical case where the driver d begins the trip after a long rest.

At the first level of the hierarchy, we assign the following initial score value to each trip:

$$s_T^1 = \frac{\nu_T^l}{\nu_d^{\max}} + \frac{r_T^l}{\omega_D}$$

Note that s_T^1 will take values between 0 and 2. Let s_{max} denote the maximum initial score over all trips. Next, we select up to 300 trips with the highest scores among those with scores in the range $[0.9s_{max},\ s_{max}]$, forming the candidate trip set C. At the second level of the hierarchy, we select a single trip for dispatch from C. To do so, we assign a second score s_T^2 between 0 and 10 to each trip $T \in C$ as follows:

- Add 3 points if trip T is not the first assigned trip for driver d during the planning horizon;
- Add 3 points if driver d begins the trip away from his domicile terminal, DOMICILE(d);
- Add 2 points if the trip returns to DOMICILE(d) at the end of the duty;
- Add 2 points if driver d begins away from domicile, and the trip dispatch time, $\max\{\text{READY}(d), \sigma_D\}$, is no greater than six hours after the driver ready time, READY(d).

We then select for dispatch a trip T with a maximum value of s_T^2, breaking ties using the initial score s_T^1. Note that giving priority to trips that reuse drivers leads to solutions that dispatch fewer drivers. Giving priority to trips with drivers away from domicile and that return drivers to domicile leads to solutions with smaller numbers of foreign beds. Finally, prioritizing short waits before dispatch for drivers away from domicile reduces layover costs.

Updating system information

Once a trip has been selected for dispatch, the system information is updated to reflect this selection. The system information to be updated includes:

- Driver information at origin of the trip and the destination of the trip. The ready time of the driver at the destination is computed as the arrival time at the destination plus the mandatory time of a long rest, τ_R.

- Load ready and cut time information of affected loads. Recall that there are precedence relations between loads. Therefore, the ready times of successor loads and the cut times of predecessor loads may have to be updated based on the dispatch times for loads to be executed in the selected trip.

Of course, all loads included in the selected trip are marked as dispatched.

A tabu mechanism

It is possible that no trip can be generated for a selected load. In this case, the load is declared tabu and unavailable for selection until some trip has been generated and the system information has been updated. After a system update, a driver has been relocated and it may therefore now be possible to generate a trip that includes the load that was declared tabu. Finally, to avoid infinite loops, a load that has been declared tabu more than five times is removed from the list of loads, and therefore will not be dispatched.

Other issues

Note that the basic heuristic described in this section does not allow any load to be dispatched after its cut time; service is treated as a hard constraint. We have also implemented versions of the heuristic that attempt to dispatch a load late by modifying its cut time (and the cut times of any successors) by fixed intervals after a load has been declared tabu more than five times; for brevity, results for these versions are not presented.

Further, it should be noted that the basic heuristic does not explicitly prevent the dispatch of loads at times that will make undispatched predecessor loads impossible to feasibly dispatch. In general, it is rare for a successor load to be dispatched prior to a predecessor, since we select loads in order of increasing cut time. We have also experimented with a version of the heuristic that prevents this possibility by simply not allowing duties to be extended with loads that have undispatched predecessors; again, for brevity, results for this version are not presented herein.

2.5 Dynamic Implementation

The solution approach we have developed for the DSLDP is intended for implementation in a dynamic environment using a rolling planning horizon, to be used to provide decision support to linehaul dispatchers. There are several important challenges that need to be addressed to realize this vision.

First, in order for the technology to be useful, solution speeds must be very fast. We set a target of 10 minutes of run time to complete the full driver schedule for a large system. Given that the technology might be used to resolve a DSLDP every few hours, we believe that this run time is acceptable.

Second, it is important to determine an appropriate planning period length and resolve interval. We believe that a three-day planning period is appropriate for large network carriers in order to allow planning of multiple trips for each driver and to minimize end-of-horizon effects. Our system assumes that information is available for all loads to be dispatched within the planning period. In reality, a large number of the loads will be known, and another large number can be predicted fairly accurately. Importantly, shipment-level forecasts are not necessary, although reasonable forecasts of trailer-level ready and cut times are important. A small set of completely unknown loads should not affect solutions drastically, and can be captured by reasonably frequent resolves. In the LTL context, resolves every few hours should be sufficient.

Third and lastly, in a dynamic implementation it is important to carefully choose which feasible prior decisions to maintain, and which to discard before resolving. For the DSLDP, we believe it is appropriate to preserve at the resolve epoch only each driver's current and next planned load dispatch (as long as they are still feasible). Another reasonable choice might be to preserve each driver's current planned trip, or preserve at least the high quality current planned trips for some subset of the drivers.

2.6 Computational Study

We now present the results of a set of computational experiments used to analyze the performance of the proposed driver scheduling technology. The basis for our computational

experiments is a month of historical data from a large national LTL carrier in the U.S. The linehaul network for this carrier contains 648 locations and 16,405 connections. The location set includes breakbulk terminals, end-of-line terminals, relay terminals, rail yards, marshalling yards, and meet-and-turn locations. While rail yards, marshalling yards, and meet-and-turn locations are not fully-equipped facilities, they will be referred as terminals for convenience in this thesis. From a driver scheduling perspective, the inclusion of rail yards, marshalling yards, and meet-and-turn locations does not create significant complications. The primary difference between these locations and terminals is that a driver may not rest to recharge his drive and duty hours at these locations. Since LTL truck traffic varies by month of the year, we chose data for a representative medium-traffic month, March 2005, to test the methodology.

There are a number of challenges that must be handled in the process of generating data sets. The LTL carrier provided the actual movements of loads, the shipments contained in these loads, and the drivers that moved the loads. Using information for the shipments contained in a load, we determine a load cut time, *i.e.*, the latest time at the origin when the load can depart and still make service for all the shipments contained in the load, by considering for each shipment the difference between the due date and the minimum time required for all downstream movement and handling activities. Ready times for loads were calculated differently for two types of historical loads. For loads that were built by loading one or more trailers, the ready time is given by the latest completion time of the trailer loading process. For loads that were built only by pup-matching existing trailers in the system, the ready time is determined by adding for each trailer the minimum upstream movement time along the path from the original trailer build location to its original ready time, and taking the maximum. It is important to note that due to certain historical decisions regarding which trailers to match into a load, it is possible that this method of generating ready and cut times may result in cases where the ready time of a load is later than its cut time. Since this occurs rarely, we arbitrarily set the cut time to be 30 minutes after the ready time for such loads. We also generate a precedence relationship between each load built by pup-matching trailers and the inbound loads moving those trailers.

2.6.1 Initial Analysis

An initial analysis of the developed driver scheduling technology was performed using 3-day instances. To create an instance, we extract from the historical data all dispatched loads with a ready time that falls within a 72-hour period. We then use our driver scheduling technology to schedule the movements of these loads (which implies scheduling of empty movements as well). A typical 3-day instance has between 15,000 and 18,000 loads and a pool of approximately 3,500 drivers is available. Table 2.1 gives the number of loads in each of the eight 3-day instances considered here. Since traffic varies significantly by day of the week, we extracted four 3-day instances $\{MW1, MW2, MW3, MW4\}$ covering Monday through Wednesday loads, and four instances $\{WF1, WF2, WF3, WF4\}$ covering Wednesday through Friday loads. Wednesday and Thursday tend to have the most traffic, while Monday traffic is light; the table reflects this fact since the end-of-week instances have approximately 20% more loads to cover.

Table 2.1: Number of loads to be dispatched in 3-day test instances for DSLDP heuristics

Instance	# Loads
$MW1$	15,275
$MW2$	14,906
$MW3$	15,023
$MW4$	15,135
$WF1$	18,392
$WF2$	17,656
$WF3$	17,666
$WF4$	17,493

We now present results that aim to validate the algorithmic design choices made for the driver scheduling scheme, *i.e.*, the use of a duty-based approach and the use of enumeration during the generation of duties. To do so, we compare the performance of three different heuristics. The first heuristic (H1) is the one described in detail in the previous section. Recall that H1 greedily selects for dispatch the outstanding load with the earliest cut time, enumerates a large tree of time-feasible driver duties that begin with the selected load, creates trips by matching drivers with duties, scores the trips, and finally selects the best

trip. The second heuristic (H2) is a simplification of the first. Heuristic H2 greedily selects the next load for dispatch using the same criterion as H1. Next, a small set of nested time-feasible driver duties are generated by recursively adding only the outbound load with maximum feasible drive time, breaking ties by choosing the one that leads to minimum wait; this method builds a single "path" of driver duties rather than a tree. This small set of duties is again matched with drivers to create trips which are scored and the best trip selected using the same approach as H1. The third heuristic (H3) is yet a further simplification which does not build a complete duty during each dispatch step. Heuristic H3 again uses the same greedy criterion for load selection, and then simply greedily selects a driver to perform the load by extending his duty (or beginning a new duty if this load is the driver's first off rest) to include the selected load. To select a driver, the heuristic first considers all drivers with partial duties and checks to see if the load can by added without violating the drive time and duty time restrictions. If so, then among the eligible drivers, the driver with the (partial) duty with maximum loaded drive time is selected, breaking ties by choosing the duty with minimum wait time. If there are no drivers with partial duties that can accommodate the selected load, then the heuristic considers drivers coming off-rest and assigns the driver with the off-rest time closest to the ready time of the selected load is chosen. If no driver is available at the origin terminal, the nearest feasible driver at another terminal is repositioned to serve the load.

Comparative results are presented in Table 2.6.1. Each heuristic was run on a 2.4 GHz Pentium Linux PC with 2 GB of memory. There are two groups of results: the first group provides averages of key performance statistics over the four Monday-Wednesday instances $MW1$–$MW4$, and the second group gives averages over the four Wednesday-Friday instances $WF1$–$WF4$. The first column indicates the heuristic used. The second column gives the percentage of loads that are dispatched. The third column shows the percentage of loads dispatched before the historical (actual) dispatch time provided in the dispatch data set. The fourth column shows the number of empty driver miles as a percentage of the number of total driver miles. The fifth column shows the number of foreign beds as a percentage of the total number of long rests. The sixth column shows the number of drivers

used. The seventh and eighth columns show the average drive time and duty time of all assigned driver trips, in hours. Finally, the last column shows the run time of the heuristic in seconds.

Table 2.2: Comparison of three greedy heuristic solution approaches for DSLDP

	%Dispatched	%Before Historical	%Empty	%Foreign Beds	# Drivers	Average Drive Time	Average Duty Time	CPU Time
H1	99.52	64.31	11.80	67.49	2,824	8.87	10.29	303.21
H2	99.49	59.28	12.98	75.76	2,921	7.25	8.35	61.31
H3	99.50	59.23	12.51	75.60	2,932	7.24	8.34	61.75
H1	99.21	61.46	9.58	65.10	3,009	8.99	10.26	224.68
H2	99.22	55.46	10.51	76.91	3,174	7.51	8.62	85.67
H3	99.24	55.65	11.62	76.90	3,187	7.52	8.64	86.12

A number of observations can be made. First and foremost, it is clear that expending the time to enumerate and evaluate a large set of feasible duties is worth the effort. Heuristic H1 produces substantial improvement over the two other heuristics in all performance indicators. The number of drivers used decreases, the average drive and duty time increase, the empty mileage percentage decreases, and the percentage of foreign beds decreases. It is important to note that the duty enumeration, when implemented carefully, did not significantly increase the computation times, which are still well below the target of 10 minutes. Perhaps surprisingly, heuristics H2 and H3 produced remarkably similar average results. Given that H2 greedily assigns a complete driver duty built to maximize loaded drive time with each load dispatch, we initially expected it to perform somewhat better than H1. In any event, the results show that a greedy search approach for DSLDP can be significantly improved by enumerating a large set of driver duties that cover a load segment to be dispatched.

It is also interesting to observe that the results vary somewhat between the Monday-Wednesday instances and the Wednesday-Friday instances, and that the differences can likely be explained by the fact that the latter include approximately 20% more loads. Each heuristic is able to reduce the percentage of empty miles for the larger instances, although the pure greedy approach H3 produces the least relative reduction. It is also worth noting that heuristics H2 and H3 both perform better relative to H1 measured by average drive and duty hours on the larger instances.

It is important to note that, in our scheme, we do not allow the loads to be delivered late. Therefore, if the load cannot be dispatched within its associated feasible time window, contrary to real life, it is not dispatched at all. We wanted to focus on on-time deliveries and maximize their percentage. The solution procedure can be modified to allow loads to be delivered late, by dynamically modifying the original cut time, when the loads cannot be dispatched after a number of times. In real life, major LTL carriers have approximately 3% of their loads late. Therefore, the 97–99% on-time dispatch rate that we will observe throughout this thesis is acceptable. It should also be noted that, a late load does not necessarily mean that all the shipments inside the load are late. It just implies that the load with the strictest cut time cannot be delivered on-time.

2.6.2 Rolling Horizon Analysis

The results of the initial analysis are promising, in the sense that the driver scheduling technology is more effective when solving 3-day instances than simpler greedy approaches. Next, we investigate the performance of the technology (heuristic H1) when it is embedded in a rolling horizon framework.

The following rolling horizon scheme was used in our computational experiments: create driver schedules for the next three days, implement the decisions for the first day, roll the horizon forward by one day and repeat. This scheme was used to determine driver schedules for a full week, which implies that seven 3-day instances were solved in the process. The instances were obtained by extracting from the historical data all dispatched loads with a ready time that falls within a 9-day period.

The computational experiment outlined above does not necessarily represent a realistic rolling horizon implementation, because of the assumption of complete load visibility in the 3-day period under consideration. In reality, of course, only a subset of these loads will be known at the time the schedule is constructed. To study the impact of reduced future load visibility, we also consider scenarios where a given percentage of loads from the second and third days of the planning horizon are selected at random and ignored. That is, a driver schedule will be constructed for a 3-day period under the assumption that all loads for the

first day are known, but only a subset of loads for the next two days is known. By varying the percentage of ignored loads on the second and third day, we gain some insight on the value of information about future loads.

We have conducted experiments for three different weeks in March of 2005. Each instance determines dispatch decisions for Monday through Sunday, using data from Monday through the following Tuesday. The results are presented in Table 2.3.

Table 2.3: Results from rolling horizon experiments for three week long instances of DSLDP

Week	%Deleted	%Dispatched	%Dispatched before Historical	%Empty	%Foreign Bed	# Drivers	Average Drive Time	Average Duty Time
	0	**98.03**	**67.62**	**12.69**	**70.75**	**2950**	**8.84**	**10.16**
	5	97.89	67.45	12.68	71.28	2881	8.86	10.2
March	10	97.74	66.87	12.50	71.33	2813	8.89	10.21
7–13	15	97.28	66.49	12.49	71.99	2741	8.89	10.2
	20	96.74	65.90	12.47	71.77	2657	8.91	10.22
	30	95.12	64.26	12.42	72.13	2507	9.02	10.32
	0	**97.97**	**68.24**	**12.72**	**71.83**	**2897**	**8.86**	**10.21**
	5	97.88	67.19	12.53	71.62	2832	8.88	10.24
March	10	97.38	66.90	12.79	71.84	2753	8.91	10.25
14–20	15	96.93	66.35	12.87	71.93	2677	8.93	10.27
	20	96.46	66.12	12.74	71.84	2618	8.96	10.29
	30	94.55	64.75	12.38	72.71	2443	8.99	10.32
	0	**98.13**	**68.05**	**13.57**	**70.29**	**2940**	**8.87**	**10.24**
	5	97.90	68.16	13.49	70.77	2882	8.9	10.25
March	10	97.65	67.92	13.24	71.15	2788	8.9	10.23
21–27	15	97.21	67.00	13.24	71.19	2719	8.93	10.24
	20	96.41	66.07	13.30	71.67	2637	8.96	10.29
	30	94.55	65.20	13.07	72.27	2468	9.04	10.35

A number of observations can be made. First and foremost, the weekly performance statistics deviate very little from the performance statistics we obtained for a 3-day period, especially in the best-case scenarios in which no future loads are ignored. The driver scheduling technology is still able to dispatch more than 98% of the loads, and it does so with roughly the same number of drivers, with about the same average drive and duty times, and with slight increases in empty mileage percentage and foreign bed percentage. Note that the increase in empty miles is to be expected, since our heuristic partially uses a reactive mechanism to bring drivers empty back to locations that are outbound-heavy; this effect is masked somewhat in the 3-day instances, where drivers are reset to the initial state for each run.

Furthermore, it seems that the driver scheduling technology is fairly robust in the sense that reduced visibility of future loads does not affect the performance significantly. The percentage of loads dispatched decreases only by about 3 percentage points when 30% of the loads are ignored on the second and third day, a fairly small change given the amount of data ignored. The empty mileage percentage and average drive and duty times do not seem to be affected at all. The most striking statistic is the reduction in the number of drivers required to move loads, which of course is due in part to the reduced number of loads dispatched. However, it should be noted that the average number of loads moved by a driver during the week increases from a little over 12 to almost 14.

2.6.3 Comparison with Historical Dispatch Data

Finally, we evaluate the relative effectiveness of the driver scheduling technology by comparing the results to statistics generated from the historical data. Although such a comparison is not without flaw, we believe it does provide some insights into the potential value of the driver scheduling technology under development. There are two primary issues to keep in mind when analyzing comparative results. First, our driver scheduling scheme has a bit more flexibility when assigning drivers to loads, since we do not explicitly account for certain union rules. Union rules come in the form of driver bids, which restrict the locations that drivers can visit. An example of a driver bid is an "ABA-bid." This indicates that the destination of a driver after he comes off rest at his domicile is restricted. More specifically, if the domicile of the driver is breakbulk A, then the first destination has to be either breakbulk B or one of its aligned satellites. Then, his next destination is back to his domicile A. Handling union rules within the driver scheduling scheme will be discussed in Chapter III. Second, in this section we assume complete knowledge of all loads that must be moved during each planning period.

Table 2.4 summarizes a comparison between the solutions generated by our driver scheduling scheme H1 and actual statistics generated from March 2005 historical dispatch data. The columns of the table summarize key performance metrics. The first column reports the average number of unique drivers dispatched during a day, a good indication of

the required driver pool size. The second column reports the average empty miles per day required by the dispatch schedule. Finally, the third and fourth columns report average drive and duty times for dispatched drivers.

Table 2.4: Comparison with historical dispatch data for a large U.S. LTL carrier using average daily statistics for March 2005

	#Drivers	Empty Miles (per day)	Average Drive Time	Average Duty Time
Historical Dispatch	3209	181,577	8.41	9.20
H1 3-day Mon-Wed	2824	122,983	8.87	10.29
H1 3-day Wed-Fri	3009	119,610	8.99	10.26
H1 Rolling	2929	146,169	8.86	10.20

The results in the table indicate that the developed solution technology is quite promising. In each experimental setting and for each performance metric, the heuristic solution outperforms the actual dispatch data, often by a substantial margin. Importantly, the schedules determined by the heuristic solution approach typically require approximately 6–10% fewer drivers, with each driving an additional 1/2 hour during each duty. Driver schedules with high drive times are not only good for the carrier, since they potentially reduce the fixed costs of drivers such as benefit compensation but also since they potentially decrease turnover costs by increasing driver job satisfaction since they are paid by the mile driven.

Note also that in the historical dispatch data, the average duty and drive times are separated by about 0.8 hours whereas for the heuristic they are separated by about 1.3 hours. Thus, the heuristic solutions include an additional half hour of waiting time. In contrast, real-world dispatchers tend to to get drivers on the road again quickly when they complete a load but do not require long rest. The heuristic appears more likely to choose a duty for a driver that includes some "wait" to find a better load candidate.

CHAPTER III

DRIVER SCHEDULING WITH UNION RULES

3.1 Introduction

There are two sets of rules that restrict how drivers may be used in LTL trucking systems. The first source of such rules is the U.S. DOT, which establishes so-called hours-of-service regulations. Designed to ensure highway safety, these regulations limit the amount of time a driver can spend both driving and on-duty between rest periods during which a driver sleeps. The second source of rules are the driver contracts negotiated between labor unions and LTL carriers. Most carriers are unionized and the negotiated contracts specify rules that restrict, sometimes severely, how drivers domiciled at different terminals in the network may be used. Chapter II has focused on developing a driver scheduling scheme capable of dispatching 15,000–20,000 loads with fleets with thousands of drivers, such that each utilized driver conforms to U.S. DOT hours-of-service regulations. Since this technology did not consider union rules, in this chapter we will remedy that by discussing how generic forms of such rules can be incorporated within the approach.

The aspects of LTL transportation systems that complicates driver scheduling described in Section 1.3.3, such as dispatch flexibility of loads, determining where drivers rest and planning for empty moves are still valid in this chapter.

3.2 Managing Driver Schedules With Union rules

In order to describe more precisely how union rules and bid rules affect LTL driver operations, we first introduce some notation. Each end-of-line terminal in the terminal network is matched with a breakbulk terminal, called *parent terminal*. Parent terminals are breakbulk terminals where consolidation with other freight occurs. While some LTL networks may feature EOLs with multiple parents that are used for freight headed in different directions, this complexity is ignored here, and it is assumed that all loads originating at an EOL are sent to parent. Shipments destined to an EOL are typically sent to its parent terminal

and then forwarded to the EOL. For convenience, we assume that a breakbulk terminal is parent of itself. The district of terminal T, denoted by $D(T)$, is the set of terminals for which T is the parent terminal, *i.e.*, $D(T) = \{t \mid T \text{ is parent of } t\}$.

3.2.1 Domiciles

As mentioned earlier, each LTL company driver has a home terminal called *domicile*, where he returns with some regularity. The large majority of network carrier drivers are domiciled at breakbulk and relay terminals. While end-of-line terminals typically do not have domiciled drivers, ones that are distant from their parent breakbulks sometimes do.

3.2.2 Driver types in unionized carriers

LTL truck drivers working for unionized carriers are typically one of two varieties. A *bid driver* is a prioritized driver, typically with seniority, whose dispatches are restricted by negotiated union rules. On the other hand, an *extra-board driver* is an unrestricted driver who can typically be dispatched to any terminal in the network at any point in time. Extra-board drivers are typically only domiciled at breakbulk or relay terminals.

The union rules that govern the usage of bid drivers have been negotiated to ensure regularity in drivers' day-to-day schedules and to improve quality-of-life measures. These rules are typically negotiated locally, and thus may vary from terminal to terminal. There are two types of restrictions: (1) location restrictions, and (2) time restrictions. Location restrictions limit the terminals that a bid driver can visit. The schedule of the driver is constrained by his domicile, his *bid type*, his *bid terminal*, and the travel time between his domicile and his bid terminal. In addition, there may be time restrictions. Time restrictions may define a *bid start time* for the driver indicating a time he is expected to be dispatched each day, or they may define a *weekly plan* that limits the days of the week on which the driver can work, or they can make the driver part of a *wheel*; drivers on the wheel get loads on first-come, first-served basis after coming off-rest. We will only consider location restrictions in this research, since they tend to have the largest impact on driver scheduling.

Since drivers are typically not salaried employees, but get paid based on the number of miles driven or the number of hours worked, unions also negotiate the work "owned" by a

terminal. A terminal may be designated as the *primary* for a set of neighboring terminals. If T_1 is the primary for a lane between two terminals T_1 and T_2, then the drivers domiciled at T_1 have priority over driver domiciled at T_2 when it comes to covering loads between T_1 and T_2, regardless of the origin of the load. Typically, for a given lane, the terminal that originates the most freight running over the lane is designated the primary. For a lane between a breakbulk and one of its aligned EOLs, the breakbulk is the primary.

For a terminal T, there is a set of terminals, denoted by $P(T)$, for which T is the primary. The drivers domiciled at T can only bid on terminals in $P(T)$ and on *district work*. District work is a special bid for a driver domiciled at a breakbulk terminal. It includes all the loads between T and terminals in $D(T)$. Note that an EOL terminal does not have district work, and the only lane on which it may be primary is the lane to its parent breakbulk terminal.

In addition to the bid terminal, location restrictions are classified by bid type. The most common location bid type is the *ABA bid*. Drivers with such a bid type are dispatched according to the following rule: the first dispatch after rest at the driver domicile (denoted by A) will be to a terminal in the district of his bid terminal (denoted by B), and the driver will be returned to A for rest after completing a duty.

Depending on the driving time between domicile A and bid terminal B, there are two subclasses of ABA bid drivers that indicate whether the driver is returned A for rest nightly, or whether the driver is only returned to A every other night:

1. *ABA laydown drivers:* If the driving time between A and B is greater than half of the maximum allowable driving time imposed by U.S. DOT, then the driver is an ABA *laydown* driver. After coming off rest at his domicile, such a driver will end his next duty at a terminal in $D(B)$. His subsequent duty after rest must return him to his domicile for his next rest. Examples of potential duties for an ABA laydown driver can be found in Figure 3.1.

2. *ABA turn drivers:* If the driving time between A and B is no greater than half of the maximum allowable driving time imposed by U.S. DOT, then the driver is an ABA *turn* driver. In this case, after coming off rest at his domicile, the driver can go to any

terminal in $D(B)$, but is required to return to his domicile A or any terminal in $D(A)$ for rest at the end of his duty. Such a driver may not be dispatched to any terminal other than the ones in $D(A)$ and $D(B)$.

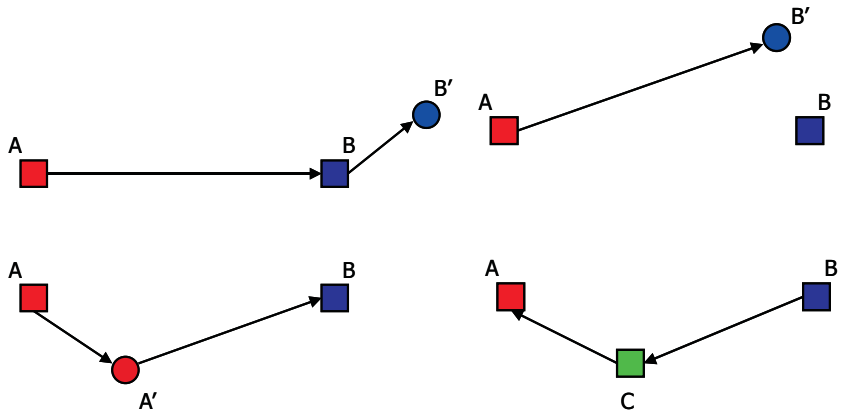

Figure 3.1: Potential duties for an ABA laydown driver. $A' \in D(A)$, $B' \in D(B)$.

It is important to note that if the carrier cannot dispatch a bid driver back to his domicile or domicile district with a loaded move to conform to the rules outlined above, the carrier will be forced to introduce empty equipment dispatches. For ABA laydown drivers, this may result in a duty consisting entirely of empty travel.

It should be clear that LTL carriers would prefer a driver fleet composed entirely of extra-board drivers, since such drivers have the flexibility to be dispatched to any terminal in the network, thus potentially maximizing the time they spend moving loads. However, the schedule regularity provided by bids is attractive to drivers, and therefore unions would like the carrier to employ as many bid drivers as possible given the pattern of available loaded movements. To manage this tradeoff, unions typically negotiate with LTL carriers a minimum percentage of bid drivers in the driver fleet. This percentage is typically 60–70 per cent for large national LTL carriers.

3.2.3 Local dispatch rules

As mentioned earlier, union rules also prioritize the drivers that can be assigned to the dispatch of a load. A driver's domicile, his status (bid versus extra-board), his bid terminal, and the primary on the load lane all impact the priority ranking of the available

drivers. Given a load to be dispatched from T_1 to T_2, available drivers at T_1 can be grouped according to priority, from highest to lowest, as follows:

1. *In-motion drivers:* In-motion drivers are drivers that have only completed a partial duty, and still have sufficient remaining drive and duty hours to execute the load (the driver's domicile and bid type do not matter).

2. *Must-go drivers:* Must-go drivers are drivers coming off rest that must be dispatched to avoid violating work rules or paying extra charges. Determining which drivers are must-go drivers depends on which terminal is the primary for the lane from T_1 to T_2. Must-go drivers are further prioritized in the following order:

   - If T_1 primary:
     (a) All drivers with domicile T_2 (note that these drivers are returning to their domicile).
     (b) ABA bid drivers domiciled at T_1 with bid terminal T_2.
   - If T_2 primary:
     (a) All ABA bid drivers with domicile T_2 (note that these drivers are returning to their domicile).
     (b) All extra-board drivers with domicile T_2 (note that these drivers are returning to their domicile).

3. *Domiciled extra-board drivers*: Domiciled extra-board drivers coming off rest are given priority over extra-board drivers not domiciled at T_1 or T_2.

4. *Non-domiciled extra-board drivers*: Extra-board drivers that are neither domiciled at T_1 nor T_2 are next in priority, and should be used before the final priority class.

5. *Should-not-go drivers:* Should-not-go drivers are those coming off rest that should be reserved for dispatch on different lanes. This class includes all bid drivers at domicile T_1 with a bid terminal other than T_2. Although such drivers typically travel on their primary lane, they may be dispatched as a last resort on other lanes.

3.3 Driver Scheduling and Load Dispatching Problem With Union Rules

The *driver scheduling and load dispatching problem with union rules* (DSLDP(U)) is an extension to the DSLDP problem introduced in Section 2.3. The problem is to determine a set of driver movement schedules for a set of loads during a given planning period, such that both U.S. DOT regulations as well as union rules are not violated. Each load is characterized by an origin terminal, a destination terminal, a ready time, a cut time, a set of preceding loads which must arrive at the origin terminal before a dispatch can take place, and a set of succeeding loads which can depart from the destination terminal only after the load has arrived there. A set of drivers is available to move the loads. Each driver has a domicile, type information that indicates his bid type or whether he is extra-board, a bid terminal if applicable, a current location at the beginning of the planning period, and a ready time at the current location. A driver schedule consists of a sequence of moves (loaded or empty) with associated dispatch times and long rest periods with start times. Sequences of moves between long rests are known as *duties*; when a driver has been assigned to a duty, we refer to the duty as a *trip*. A feasible driver schedule must satisfy U.S. DOT regulations governing maximum drive hours and duty hours between long rest periods. Moreover, the dispatches must comply with the driver class prioritization defined by the local dispatch rules. The objective of the DSLDP(U) is to find a set of driver schedules with minimum total operating costs, such that all loads are dispatched between their ready and cut times.

Formally, we use the following notation to model the driver scheduling problem:

- $\mathcal{T}$ The set of all terminals.
- $\mathcal{C}$ The set of all connections.
- $\mathcal{L}$ The set of all loads.
- $\mathcal{D}$ The set of all drivers.

The following parameters are used to model U.S. DOT driver regulations:

- τ_V The maximum number of drive hours between rests.
- τ_U The maximum number of duty hours (including waiting) between rests.
- τ_R The minimum duration (hours) of a rest.

In the remainder of this paper, we use current U.S. DOT driver regulations: $\tau_V = 11$, $\tau_U = 14$, and $\tau_R = 10$.

Each load $\ell \in \mathcal{L}$ has the following attributes:

$\text{ORIG}(\ell)$	The origin terminal.
$\text{DEST}(\ell)$	The destination terminal.
$\text{READY}(\ell)$	Earliest dispatch time.
$\text{CUT}(\ell)$	Latest dispatch time.
$\text{PREC}(\ell)$	The set of immediate predecessor loads.
$\text{SUC}(\ell)$	The set of immediate successor loads.

Note that an immediate predecessor load $p \in \text{PREC}(\ell)$ must have $\text{DEST}(p) = \text{ORIG}(\ell)$, and an immediate successor load $s \in \text{SUC}(\ell)$ must have $\text{ORIG}(s) = \text{DEST}(\ell)$.

Each driver $d \in \mathcal{D}$ has the following attributes, representing his state at the beginning of the planning period:

$\text{DOMICILE}(d)$	The domicile.
$\text{TYPE}(d)$	Type of the driver.
$\text{BIDTERM}(d)$	Bid terminal for bid drivers.
$\text{LOCATION}(d)$	The current location.
$\text{READY}(d)$	Earliest dispatch time at the current location.
$\text{MAXDRIVE}(d)$	Remaining drive time.
$\text{MAXDUTY}(d)$	Remaining duty time.

Note that driver d may be enroute to the current location $\text{LOCATION}(d)$ at the beginning of the planning period. In this case, and in the case where a driver is waiting (not resting) at the current location, $\text{MAXDRIVE}(d)$ and $\text{MAXDUTY}(d)$ may be less than the U.S. DOT maximums. For the case in which the driver is resting at the beginning of the planning period, $\text{MAXDRIVE}(d) = \tau_V$ and $\text{MAXDUTY}(d) = \tau_U$. The parameter $\text{TYPE}(d)$ indicates if the driver d is a bid driver or an extra-board driver; in this research, we assume that all bid drivers are the ABA type. For bid drivers, $\text{BIDTERM}(d)$ indicates the bid terminal.

Each connection in $\mathcal{C}$ links two terminals $a, b \in \mathcal{T}$ and has the following attributes:

$\textsc{cost}(a, b)$	The cost (in dollars) of driving between terminals a and b.
$\textsc{time}(a, b)$	The drive time (in hours) between terminals a and b.
$\textsc{rtime}(a, b)$	The run time (in hours) between terminals a and b.
$\textsc{primary}(a, b)$	The primary terminal (if applicable) on the lane from a to b.

The set of connections defines the linehaul network. As mentioned earlier, if two terminals are connected, the drive time between them is no greater than τ_V, and there is frequent transportation between the two terminals. Depending on the drive time between two terminals, a driver may need to rest along the way; referred to as short rests. These short breaks are incorporated in the run time and count towards duty time. Note that $\textsc{rtime}(a, b) \leq \tau_U$ for all connected terminals. Not every terminal-to-terminal connection defines a lane with loaded moves, and further not all lanes have an associated primary terminal. When applicable, $\textsc{primary}(a, b)$ represents the primary terminal for the lane from a to b and is equal to either a or b.

In addition to the cost incurred while driving between two terminals, a fixed cost is incurred for a driver resting at a location other than his domicile (foreign bed cost).

3.4 Solving DSLDP(U)

The dynamic driver scheduling scheme described in Section 2.4 is an iterative procedure that generates feasible driver assignments that conform to U.S. DOT regulations. At each iteration, the algorithm selects a load to be dispatched next, called *base load*. Next, it generates a set feasible duties, *i.e.*, sequences of loads starting with the base load. Afterwards, a set of trips are created by coupling the duties with drivers that can feasibly execute the duty. Finally, each trip is scored based on the objectives and one of them is selected to be actually dispatched. This procedure is repeated until all loads are dispatched. In this section, we describe how the driver scheduling scheme presented in Section 2.4 is enhanced to properly handle the added complexity of union rules.

As noted in the previous section, in-motion drivers have the highest priority among the drivers. The driver scheduling scheme always generates a complete duty for a driver, *i.e.*,

the driver is sent to rest at the end of the assigned duty. Therefore, during each iteration of the approach, no drivers in the system are in-motion except the driver assigned to the duty in the currently selected trip. Thus, the duty extension mechanism in the approach and the fact that executed trips always end with a driver put to rest guarantee that the approach never violates the priority given to in-motion drivers.

Once the next load to be dispatched has been selected by the approach, it is not difficult to group the available drivers into the prioritization hierarchy implied by the dispatch rules: must-go drivers, domiciled extra-board drivers, non-domiciled extra-board drivers, and should-not-go drivers. To properly account for this prioritization, the driver scheduling scheme is extended to generate duties and trips for each group in order, and to dispatch the best generated trip within a priority group. If no trips are generated for a priority group, the approach moves on to the next group. Since the drivers within each priority group may face additional restrictions on how they can move between terminals, the duty generation mechanism of the approach is enhanced to account for these rules.

A high level description of the extended driver scheduling scheme is given in Algorithm 3.1.

Algorithm 3.1 High level overview of the driver scheduling scheme with union rules.

```
while there exist undispatched loads do
  Select a load to be dispatched next.
  for each driver group in the order described in Section 3.2.3 do
    Generate a set of duties that include the selected load as the first loaded move of the
    trip
    Generate a set of trips by matching drivers with duties
    if any trip generated then
      Assign a score to each trip indicating its desirability
      Select a trip with the highest score
      Break for loop
    end if
  end for
  Update system information
  if dispatch driver is type ABA laydown then
    Create a return trip for dispatch driver.
  end if
end while
```

Generating Duties Based on Driver Class

To be able to generate feasible duties for all driver classes, we need to be able to limit both the intermediate terminals a driver can visit and the terminal the driver ends up in at the end of his trip. We denote these two sets by $\mathcal{I}$ and $\mathcal{E}$ respectively. Clearly, $\mathcal{E} \subset \mathcal{I}$, because a driver cannot end his duty in a terminal that he cannot visit. Naturally, both of these sets can be equal to the set of all terminals, $\mathcal{T}$.

We generate all time-feasible duties as described in Section 2.4 taking U.S. DOT regulations and the timing aspects into consideration. However, we need two modifications to the duty generation scheme to enable us generate feasible duties for bid drivers.

1. When expanding a base duty with a candidate load ℓ, we make sure that $\text{DEST}(\ell) \in \mathcal{I}$.
2. If $\text{DEST}(\ell) \in \mathcal{E}$ then, we mark the duty as *saved* otherwise we mark it as *temporary*.

Only the *saved* duties are feasible and go to the scoring phase. We generate *temporary* duties because such a duty can become feasible with the addition of some loads to the end of the duty.

It is possible that expanding duties in this way may lead the search to some part of the network where it is not possible to return to any one of the terminals in $\mathcal{E}$ and waste significant computation time. To avoid such a situation, we limit the expansion of a duty as follows: We check if the duty can still be extended (possibly with an empty move) to return to one of the terminals in $\mathcal{E}$ with the remaining drive time. If such a return is not possible, we end the expansion of duty immediately.

Handling Different Driver Classes

Let us assume that load ℓ is selected as the base load to be dispatched.

The first class of drivers that is eligible for dispatch is the set of must-go drivers. Among the must-go drivers foreign drivers going home have the highest priority followed by ABA bid drivers. If $\text{ORIG}(\ell)$ is the primary on the lane associated with load ℓ, then the bid drivers with bid terminal $\text{DEST}(\ell)$ should be dispatched first from among the bid drivers. If these drivers are ABA laydown drivers, then duties are generated by setting $\mathcal{I} = \mathcal{T}$ and

$\mathcal{E} = D(\text{DEST}(\ell))$. If these drivers are ABA turn drivers, then duty generation is done by setting $\mathcal{I} = D(\text{ORIG}(\ell)) \cup D(\text{DEST}(\ell))$ and $\mathcal{E} = D(\text{ORIG}(\ell))$.

The next two classes of drivers eligible for dispatch are the set of domiciled extra-board drivers at $\text{ORIG}(\ell)$ and non-domiciled extra-board drivers. Drivers from both classes can travel anywhere in the network and their duties are generated by setting $\mathcal{I} = \mathcal{T}$ and $\mathcal{E} = \mathcal{T}$.

The last class of drivers eligible for dispatch is the set of should-not-go drivers bidding on some terminal $T' \neq \text{DEST}(\ell)$. For these drivers, duties are generated by setting $\mathcal{I} = \mathcal{T}$ and $\mathcal{E} = D(T')$.

Expanding short duties

Bid drivers are restricted in terms of where they can be dispatched to. If we limit duty generation with only loaded moves, it is possible that the resulting duties will be short, and more importantly very few, if not none, feasible duties will be generated. This is especially true for ABA turn drivers. Therefore, in the duty generation step, if a duty cannot be expanded with a load and the total drive time of the duty is below a threshold, then an empty move to each one of terminals in $\mathcal{E}$ is appended to the end of the duty if it does not violate the U.S. DOT hours-of-service requirements. The duty expansion continues after the empty move. Since the destination of the added empty move is in $\mathcal{E}$, we are guaranteed not to violate bid requirements of the drivers and the resulting duty is definitely feasible. If we can further extend the duty with loaded moves, then the utilization of the driver also increases.

Limiting the number of duties generated

We generate time-feasible duties using the depth-first recursive procedure described in Section 2.4, and similarly, we will sometimes truncate the recursion to limit the number of duties generated. For bid drivers, the truncation may be too restricting, since some of the duties generated are *temporary* and will not go into the scoring phase. Both to increase the number of *saved* duties and the quality of the duties generated, we prioritize terminals considered for expansion. The loads that end at the drivers bid terminal, his domicile, or any terminal in $\mathcal{E}$ have higher priority than the remainder of the loads. Note that the

extra-board drivers are not effected, since every duty generated for them is feasible, and therefore *saved*.

Trip generation

Trip generation is same as the previous chapter, except for a minor difference in the set of drivers considered for "bring in" with an empty move. If no trips are generated for drivers at the origin terminal of the selected load, similar to previous chapter, duties are created for drivers that start at another terminal. However, only extra-board drivers are considered for "bring in" a driver with an empty move, and bid drivers are not eligible for this procedure.

Return trip for ABA laydown driver

After an ABA laydown driver is dispatched from his domicile terminal A, such a driver always rests at a terminal in $D(B)$. Since, the driver must return to his domicile in his subsequent duty, the algorithm immediately generates the return trip for such a driver.

We consider a hierarchy of possibilities for the return trip. First, a set of duties are generated with base load as the one with the earliest cut time and with destination to the domicile of the driver. If such a load that the driver can feasibly be dispatched with does not exist, as the next option, we consider moving the driver with a load to an intermediate terminal and generating duties that end at his domicile. However, the set of intermediate terminals is limited to the ones that the duty can feasibly be extended to his domicile. If still no duties are generated, then we move the driver empty. If the driver is currently not at his bid terminal, but at an end-of-line in the district of his bid terminal, we consider moving the driver empty to his bid terminal. The chances for generating a duty that ends at the domicile are higher at his bid terminal. Finally, if no duties are generated at this point, the driver is dispatched empty to his domicile from his current location.

3.5 Computational Study

We will now present a computation study demonstrating how the solution produce described in the previous section performs. We will use the same 3-day instances from the previous chapter, where Table 2.1 shows the number of loads in each instance.

In these experiments, $\frac{2}{3}$ of the total number of drivers are assumed to be bid drivers. Then, the bid drivers were allocated to bid terminals based on the outbound *flow* from each terminal, where flow between two terminals is defined to be the number of loads times the drive time of each load. Computational experiments were run on 2.4 GHz Pentium Linux PC with 2 GB of memory. Table 3.1 shows the performance statistics for each instance with averages among MW and WF instances.

Table 3.1: Performance statistics for 3-day instances using DSLDP(U) heuristic.

	%Dispatched	%Before Historical	%Empty	%Foreign Beds	#Drivers	Average Drive Time	Average Duty Time	CPU Time
$MW1$	97.29	64.18	8.41	62.25	3150	9.00	10.04	81.23
$MW2$	97.75	65.29	8.22	62.16	3117	8.88	9.97	64.74
$MW3$	97.26	64.43	8.59	62.57	3110	8.94	10.00	68.54
$MW4$	97.40	65.31	7.88	61.28	3136	8.95	9.99	67.57
Avg(MW)	97.42	64.80	8.28	62.07	3128	8.94	10.00	70.52
$WF1$	97.10	60.72	7.82	60.63	3224	9.19	10.20	94.58
$WF2$	97.42	60.60	7.84	60.20	3132	9.22	10.19	80.09
$WF3$	97.48	60.07	8.00	59.99	3143	9.21	10.19	84.75
$WF4$	97.00	60.66	8.65	59.82	3161	9.17	10.19	83.64
Avg(WF)	97.25	60.51	8.08	60.16	3165	9.20	10.19	85.77

Compared to the results in 2.6.1, we notice that the dispatch ratio is 2% lower. This is most likely because of how we determine the number of bid drivers on each lane. Note that $\frac{2}{3}$ bid driver ratio is an estimate, and it is applied to all terminals in the network. We also assume that all drivers at the domicile at the beginning of the planning horizon, which may cause problems with return trips of bid drivers in the early stages of the planning horizon.

We see a significant decrease of 4% in the empty percentage. Although this decrease can be attributed to the decrease in the percentage dispatched, this is mostly due to the fact that bid drivers have more regularity in their schedules, and they are back to their domiciles either for every rest, or for every other rest. The same reasoning applies to the decrease in the percentage of foreign beds and to the slight increase in average drive time. Not surprisingly, in the experiments with union rules, 150–200 more drivers are used. Union rules restrict where drivers can be dispatched to and more drivers are needed to cover the same number of loads.

CHAPTER IV

DRIVER FLEET SIZING AND ALLOCATION

4.1 Introduction

Perhaps the most important tactical questions faced by LTL carriers regarding their driver fleets are determining the terminals at which to domicile drivers, determining how many drivers to locate at each of these domiciles, and determining how many drivers at each domicile should bid on certain bid terminal regions. These driver domiciling decisions interact directly with operational driver scheduling, since domiciling decisions essentially determine where drivers will be available for use. Thus, the rules (both regulatory and union) that restrict driver scheduling decisions should play a significant role in driver fleet sizing and allocation.

Driver allocation can have a significant impact on the performance of LTL linehaul operations. If too many drivers are allocated to a domicile or the distribution of drivers to various bid terminals is not appropriate, there may be too little work for the drivers resulting in poor utilization and increased costs. If, on the other hand, there are too few drivers allocated to a domicile or an incorrect allocation to bid terminals, then the carrier may not be able to cover the outbound loads within their ready and cut time windows resulting in poor on-time performance and a reduced level of customer service.

The primary complication of driver fleet sizing and allocation is the fact that an individual driver may cover loads during the course of his dispatch schedule that are neither outbound from nor inbound to his domicile terminal. Therefore, such drivers can be thought of as "helping" drivers domiciled at other terminals to serve all of their loads. While bid drivers may provide a limited amount of such helping activity, drivers without bids, such as the extraboard drivers introduced in Chapter III, have the potential to serve a large variety of loaded dispatches from many non-domicile terminals since they have very few dispatch restrictions. Given the complexity of the driver scheduling and load dispatching problem,

it is very difficult to predict either the level or location of such helping interactions without a detailed simulation of operational dispatch decisions. Thus, it seems likely that simple driver allocation rules may be overly simplistic and may lead to ineffective decisions.

In this chapter, we focus on three tactical questions related to driver fleet sizing and allocation to terminals:

- How many drivers should be domiciled at each terminal?
- What fraction of these drivers should be bid drivers?
- What should be the bid terminal for each bid driver?

The methodology we propose to develop effective answers to these questions utilizes the driver scheduling tool described in Chapter III as a dispatch simulator. Given a set of drivers with known domiciles and bid information, recall that the scheduling tool determines effective driver schedules and load dispatches satisfying all rules and restrictions, generating empty movements as necessary. In this chapter, the scheduling tool is used to evaluate a current driver domicile state within a scheme that iteratively adjusts the state to improve the percentage of loaded movements covered by driver schedules.

The primary contributions of this chapter can be summarized as follows:

- We develop and describe a tactical driver domiciling methodology that can be used to determine the total required driver fleet size, the fraction of the driver fleet to domicile at each terminal, and the bid driver percentage and bid terminals for each domicile location, while maintaining a fixed system-wide total fraction of bid drivers; and
- We present a computational study using truck movement data representative of operations at a major U.S. LTL carrier that demonstrates the effectiveness of our domiciling methodology, and further shows that typical union rules may lead to increases in the required driver fleet of up to 50%.

The remainder of this chapter is organized as follows. In Section 4.2, we describe the driver domiciling methodology in detail. Then, Section 4.3 demonstrates the utility of the

domiciling methodology in a computational study based on real-life loaded movement and terminal network data from a large U.S. LTL carrier.

4.2 Driver Fleet Sizing and Allocation Methodology

The driver fleet sizing and allocation methodology we propose is based on an iterative scheme that requires as inputs a set of required loaded dispatches, a fixed terminal network with a set of terminals $\mathcal{M} \subseteq \mathcal{T}$ that can be used as domiciles, and a minimum ratio ρ of bid drivers to all drivers in the system. At its conclusion, the output of the scheme is a total driver fleet size, an allocation of the driver fleet to domicile terminals, and for each domicile terminal the fraction of allocated drivers assigned to each bid terminal as well as to the extra-board type.

The scheme is not designed to be completely automated, and requires user intervention at two distinct decision points. Currently, the scheme uses two phases: in the first phase, a fleet size and allocation of bid drivers is determined, and in the second phase, an allocation of extra-board drivers is determined for a fleet size no greater than a limit imposed by ratio ρ. During each phase of the scheme, an initial allocation of some relatively small set of drivers that is certain not to be capable of serving all loaded dispatches is determined via a simple allocation rule. Then, the driver scheduling tool is used to plan best schedules for the drivers in this set. Using the loads that remain unserved, the simple allocation rule is used again to allocate (and re-allocate) a small set of m_{new} new drivers along with drivers that were not dispatched during the previous scheduling run. This iterative process of scheduling and allocating is repeated in each of the two phases until the user decides that enough drivers have been added, or until an upper bound on the number of drivers (for example, the one implied by ρ during the extra-board phase) has been reached.

Algorithm 4.1 gives a high level description of the driver fleet sizing and allocation scheme. In the phase I portion of the scheme, only bid drivers are added to the system during each iteration, beginning with m_{init} drivers. Since bid drivers have priority, treating them separately guarantees that the domiciling approach is not biased by the presence of extra-board drivers who might be assigned to "bid work" after all bid drivers have been

used; such an effect is especially likely in the initial iterations when the driver fleet is still small relative to the number of available loads. In practice, the scheme is iterated until a large fleet of bid drivers has been allocated. Using system performance statistics, the user then decides on an appropriate bid driver fleet size and allocation from the available iterations. The primary performance statistics to consider are the percentage of loaded movements that are served, and driver performance statistics such as average drive time, average duty time, and average number of trips per driver per week. As more and more drivers are added to the system, the user will notice diminishing returns with respect to percentage of loaded movements served (*e.g.*, the increase in percentage served per driver added will decrease), and the driver performance statistics may begin to suffer. Judgement is then used to select an appropriate bid driver fleet size, m_{bid}.

In phase II, extra-board drivers are added. Using only the loads that remain unserved by bid drivers after phase I, the process is continued by adding during each iteration m_{new} extra-board drivers allocated to terminals using a simple rule. In phase II, the iterative allocation process concludes when the ratio of bid drivers to all drivers exceeds ρ.

Algorithm 4.1 High level description of the driver allocation algorithm.

$m_{bid} \leftarrow m_{init}\,,\ m_{eb} \leftarrow 0$

{Start Phase I}

repeat

Run Algorithm 3.1

Calculate remaining bid work for all terminals

$\mathcal{D}_a = m_{new}$ new bid drivers $\cup$ unassigned bid drivers in previous iteration

Allocate $\mathcal{D}_a$ to domiciles and to bid terminals

$m_{bid} \leftarrow m_{bid} + m_{new}$

Reset all load dispatches

until (No additional loads covered **or** No significant increase in marginal percentage load coverage)

User selects appropriate bid driver fleet size m_{bid} and corresponding driver allocation from discrete choices generated in Phase I

{Start Phase II}

Remove all bid drivers and loads covered by them.

repeat

Run Algorithm 3.1

Calculate remaining work for all terminals

$\mathcal{D}_a = m_{new}$ new extra-board drivers $\cup$ unassigned extra-board drivers in previous iteration

Allocate $\mathcal{D}_a$ to domiciles.

$m_{eb} \leftarrow m_{eb} + m_{new}$

Reset all load dispatches

until $(m_{eb} \geq \frac{1-\rho}{\rho} m_{bid})$

4.2.1 Calculating Remaining Work

We define *work* for a terminal to be the total drive time required by the loads that should be covered by the drivers domiciled at the terminal. Let us denote the set of drivers

that is added at each iteration by $\mathcal{D}_a$; recall that $\mathcal{D}_a$ includes both the set of new drivers added to the system, as well as all drivers that were not dispatched in the previous iteration. In both phases, each iteration begins with calculating the remaining work for each terminal. Based on the remaining work, the drivers in $\mathcal{D}_a$ are allocated to domiciles using a simple proportional allocation rule. Since we are adding bid drivers in phase I, we also determine the bid terminal for each new driver, again using a proportional rule based on remaining work.

We now describe in detail how to calculate the remaining undispatched work for terminals. Bid drivers domiciled at T_d are responsible for covering loads between T_d and each terminal in $P(T_d)$; note that this includes loads moving in both directions on these lanes during the entire planning horizon. Of the drivers domiciled at T_d, those that bid on $T_b \in P(T_d)$ are responsible for covering all loads moving back and forth between T_d and any terminal in $D(T_b)$. Let $n_u(T_1, T_2)$ denote the number of unassigned loads from terminal T_1 to T_2 and let $W_{bid}(T_d, T_b)$ denote the remaining amount of bid work that a driver domiciled at T_d with bid terminal T_b should cover, *i.e.*,

$$W_{bid}(T_d, T_b) = \sum_{T' \in D(T_b)} \big(n_u(T_d, T') * \textsc{time}(T_d, T')\big) + \sum_{T' \in D(T_b)} \big(n_u(T', T_d) * \textsc{time}(T', T_d)\big)$$

The total unassigned bid work for a domicile $T_d \in \mathcal{M}$ is

$$\overline{W}_{bid}(T_d) = \sum_{T' \in P(T_d)} W_{bid}(T_d, T'),$$

and the total unassigned work is

$$\widetilde{W}_{bid} = \sum_{T_d \in \mathcal{M}} \overline{W}_{bid}(T_d).$$

The logic for calculating the remaining work in phase II is similar to phase I. However, the definition of what constitutes work for a extra-board driver is slightly different.

The primary work responsibility of an extra-board driver domiciled at T_d is comprised of all the outbound loads from his domicile that are not covered by bid drivers. A reasonable approach then is to locate extra-board drivers in phase II based on the total outbound drive time required by unassigned loads for each terminal. Let $N(T_d)$ be the set of destination

terminals for which there are unassigned loads originating at terminal T_d. First, we calculate the amount of unassigned work from T_d to each $T_n \in N(T_d)$:

$$W_{eb}(T_d, T_n) = n_u(T_d, T_n) * \text{TIME}(T_d, T_n).$$

The total unassigned work for a domicile $T_d \in \mathcal{M}$ is then

$$\overline{W}_{eb}(T_d) = \sum_{T' \in N(T_d)} W_{eb}(T_d, T'),$$

and the total unassigned work is

$$\widetilde{W}_{eb} = \sum_{T_d \in \mathcal{M}} \overline{W}_{eb}(T_d).$$

4.2.2 Allocating Drivers to Terminals and Bids

Each iteration, the driver set $\mathcal{D}_a$ is allocated to domicile terminals and bid terminals using a simple allocation rule proportional to unassigned work. Therefore, the number of new bid drivers that a terminal receives in phase I is given by

$$m_{newbid}(T_d) = \left\lceil \frac{\overline{W}_{bid}(T_d)}{\widetilde{W}_{bid}} * |\mathcal{D}_a| \right\rceil.$$

Next, we need to determine the bid terminal for each bid driver allocated to T_d. The number of drivers with bid terminal $T_b \in P(T_d)$ will be proportional to the unassigned work between T_d and T_b:

$$m_{newbid}(T_d, T_b) = \left\lceil \frac{W_{bid}(T_d, T_b)}{\overline{W}_{bid}(T_d)} * m_{newbid}(T_d) \right\rceil.$$

In phase II, the allocation of extra-board drivers to terminals is similarly as follows:

$$m_{neweb}(T_d) = \left\lceil \frac{\overline{W}_{eb}(T_d)}{\widetilde{W}_{eb}} * |\mathcal{D}_a| \right\rceil.$$

In phase II, we do not need to decide on bid terminals because all the new drivers are extra-board and thus do not have a bid terminal.

4.2.3 Initialization

We start Phase I with a number of bid drivers m_{init} in the system to save computational run time. We decided to use an initial number of drivers that is approximately equal to half

the number of bid drivers we expect the system to require. Initial allocation of drivers is done by considering all loads as undispatched and using the formulas described in Section 4.2.1 and 4.2.2.

4.2.4 Combining Results from Different Load Set Inputs

The driver fleet sizing and allocation scheme described above is designed to use a single set of loads over a fixed planning period to determine an allocation. Since LTL operations are naturally divided into week-long periods, it is appropriate to use load sets that consist of a week's worth of loads. However, since a week is a relatively short period from a tactical planning perspective and since the loads that must be served will likely vary from week to week, an effective fleet sizing and allocation scheme should use multiple load sets to avoid any bias from a particular week's worth of loads.

To use multiple load sets, it is good practice to use planning periods of equal length. An appropriate method is to determine a fleet size m_{bid} and a resultant m_{eb} by analyzing the results of the phase I bid iterations over the multiple periods, and selecting a number of bid drivers m_{bid} that leads to satisfactory system performance statistics for each of the load sets. Given the fixed value m_{bid}, the phase II extra-board iterations can then be executed for each of the load sets.

The result of this process will be an allocation of m_{bid} bid drivers to domiciles and bid terminals and an allocation of m_{eb} extra-board drivers to domiciles that differs for each of the load set inputs. In order to determine a final allocation, therefore, we propose to combine the allocation results from the runs corresponding to each of the multiple load set inputs using regression models with the objective of minimizing the total absolute deviation of the final allocated number of drivers from the allocated numbers in each of the runs.

For bid drivers, let

- $m^i_{bid}(T_d, T_b)$ be the number of bid drivers assigned to terminal T_d with bid terminal T_b for the run corresponding to load set i, and
- $\widetilde{m}_{bid}(T_d, T_b)$ be the decision variable for the final number of drivers at T_d with bid terminal T_b.

We then solve the following optimization problem to determine the final bid driver allocation:

$$\min \sum_{T_d \in \mathcal{M},\ T_b \in P(T_d)} \sum_i \left| \widetilde{m}_{bid}(T_d, T_b) - m^i_{bid}(T_d, T_b) \right|$$

subject to

$$\sum_{T_d \in \mathcal{M},\ T_b \in P(T_d)} \widetilde{m}_{bid}(T_d, T_b) = m_{bid}$$

For extra board drivers let

- $m^i_{eb}(T_d)$ be the number of extra board drivers assigned to domicile T_d for load set i, and
- $\widetilde{m}_{eb}(T_d)$ be the decision variable for the number of extra board drivers domiciled T_d.

We then solve the following problem to determine the final extra-board driver allocation:

$$\min \sum_{T_d \in \mathcal{M}} \sum_i \left| \widetilde{m}_{eb}(T_d) - m^i_{eb}(T_d) \right|$$

subject to

$$\sum_{T_d \in \mathcal{M}} \widetilde{m}_{eb}(T_d) = m_{eb}$$

4.3 Computational Study

We test the driver fleet sizing and allocation scheme described in the previous sections using load sets representing three consecutive weeks of real loaded dispatch data for a major national LTL company. Each week begins on a Monday at 12 a.m., and concludes on a Sunday at 11:59 p.m. The terminal network through the loads move is identical to the network considered in the computational experiments presented in Chapters II and III, including all of the primary lane definitions. Drivers are assumed to be permitted to be domiciled at any breakbulk, relay, or EOL terminal; again, if drivers are domiciled at an EOL, they must be bid drivers and may only bid on the parent breakbulk.

Table 4.1 shows the number of loads to be dispatched in each of the three weeks.

Table 4.1: Number of loads to be dispatched in week long test instances for driver allocation schemes.

Instance	# Loads
W1	23,289
W2	23,037
W3	22,234

We conduct three computational experiments. The first experiment is designed to determine whether the iterative driver allocation scheme described in Algorithm 4.1 generates better solutions for a fixed number of bid and extra-board drivers m_{bid} and m_{eb} than alternative allocation schemes that use simple proportional allocation rules in a more naive manner. The results demonstrate that the iterative scheme substantially outperforms simpler rules.

The second experiment investigates the system performance impacts of varying the number of drivers while keeping ρ (the percentage of bid drivers) constant. Such an experiment demonstrates the potential for the technology to aid an LTL carrier in making decisions regarding both the driver fleet size and the allocation of this fleet to domiciles and bid terminals.

In the final experiment, we use the driver fleet sizing and allocation scheme in the case where $\rho = 0$. By applying only phase II of the approach, we generate a near-optimal fleet size and allocation for a system that includes only extra-board drivers. Having such an all extra-board baseline, we assess the impact of union rules on the operations of an LTL carrier, *i.e.*, the additional costs incurred.

4.3.1 Experiment 1: Comparison of Different Driver Allocation Algorithms

In this computational experiment, we compare two schemes for allocating 3,000 drivers with $\rho = 2/3$. In the *base allocation scheme*, we allocate the entire driver fleet in two steps, both using proportional allocation rules. To do so, we first combine all of the loads from the separate instances into a single large load set. Next, we calculate the total bid work per lane, per terminal and for the total system using the formulae in Section 4.2.1 for W_{bid}, $\overline{W}_{bid}$ and $\widetilde{W}_{bid}$, and use the proportional allocation formulae in Section 4.2.2

with $|\mathcal{D}_a| = 2,000$ to determine domiciles and bid terminals for the bid drivers. Finally, we determine a proportional allocation of the 1,000 extra-board drivers to domiciles by calculating the total outbound work for each terminal using the formulate in Section 4.2.1 for W_{eb}, $\overline{W}_{eb}$ and $\widetilde{W}_{eb}$, and using the allocation formulae in Section 4.2.2 with $|\mathcal{D}_a| = 1,000$.

The second allocation scheme uses the approach summarized in Algorithm 4.1 for a fixed fleet size of $m_{bid} = 2,000$ and $m_{eb} = 1,000$ to determine allocations for instances W1, W2, and W3. Then, the optimization formulations given in Section 4.2.4 are used to generate a final best allocation.

Given the final allocations from each scheme, the driver scheduling tool from Chapter III is used to create best driver schedules for each of the three load set instances W1, W2, and W3 to generate the results for comparison. Table 4.2 summarizes and compares the primary driver performance statistics under each of the driver allocation schemes. The third column reports the number of drivers assigned for that week. The fourth column reports the percentage of loads that are dispatched. The fifth column reports the number of empty driver miles as a percentage of the number of total driver miles. The sixth column reports the number of foreign beds as a percentage of the total number of long rests. The seventh and eighth columns report the average drive time and duty time of all assigned driver trips, in hours. Finally, the last column reports the number of tours executed during the week per dispatched driver.

Table 4.2: Comparison of two variants of driver allocation schemes.

Scheme	Date	# Drivers Assigned	% Loads Dispatched	%Empty	%Foreign Beds	Average Drive Time	Average Duty Time	Tours per driver
Base Allocation	W1	2118	82.99	7.45	43.47	9.46	10.66	4.19
	W2	2144	80.51	8.24	45.08	9.38	10.55	4.01
	W3	2117	83.03	7.57	43.82	9.42	10.58	4.08
Algorithm 4.1	W1	2621	98.52	9.15	55.23	9.37	10.62	4.28
	W2	2581	97.05	9.35	56.53	9.21	10.46	4.20
	W3	2589	98.66	8.82	52.40	9.39	10.64	4.12

The difference in performance is surprisingly large. With the driver allocation that results from the simple proportional driver allocation scheme, only about 82% of loads are dispatched and less than $\frac{2}{3}$ of the drivers are being used. On the other hand, with the driver

allocation that results from the iterative driver allocation scheme, about 98% of loads are dispatched with about $\frac{9}{10}$ of the drivers. These results demonstrate that capturing the complex interactions between available loads and the drivers moving through the terminal network requires careful analysis. The iterative allocation scheme essentially performs such analysis by simulating and evaluating the system after adding a set of new drivers in each iteration and adjusting the allocations based on the information obtained.

4.3.2 Experiment 2: Varying the Number of Drivers with Constant Bid Driver Ratio

As mentioned in the introduction, one of the key tactical decisions an LTL carriers must make is the number of drivers to hire and where to locate them in the linehaul network. This is a difficult decision for many reasons. Freight flows are seasonal, so driver needs are not constant over time. Furthermore, assessing the amount of freight that can be handled for a given number of drivers and a given driver allocation is nontrivial. In this computational experiment, we analyze the impact of varying the number of drivers in the system while keeping the bid driver ratio, ρ constant. For each fleet size setting, we use the iterative driver allocation scheme in Algorithm 4.1 to decide where to locate drivers for each of the weekly load instances W1, W2, and W3. The optimization formulations from Section 4.2.4 are again used to determine a final best allocation from these three weekly runs, and the driver scheduling technology is used to create the final schedule and the performance results. Table 4.3 presents these results for cases with 3,300, 3,000 and 2,500 total drivers.

Table 4.3: Performance statistics with various number of drivers using the iterative driver allocation scheme.

Driver Allocation	Date	# Drivers Assigned	% Loads Dispatched	%Empty	%Foreign Beds	Average Drive Time	Average Duty Time	Tours per driver
2200 Bid &	W1	2772	98.69	9.11	53.17	9.34	10.61	4.05
1100 Extra-	W2	2790	97.83	9.26	54.43	9.16	10.43	3.99
board	W3	2726	98.66	9.09	52.40	9.34	10.58	3.99
2000 Bid &	W1	2621	98.52	9.15	55.23	9.37	10.62	4.28
1000 Extra-	W2	2581	97.05	9.35	56.53	9.21	10.46	4.20
board	W3	2589	98.66	8.82	52.40	9.39	10.64	4.12
1700 Bid &	W1	2244	94.70	9.51	56.72	9.46	10.66	4.75
800 Extra-	W2	2269	93.95	9.91	58.75	9.33	10.54	4.62
board	W3	2249	95.41	9.21	55.88	9.46	10.67	4.61

We observe that only slightly higher dispatch rates can be achieved with 3,300 compared to 3,000 drivers. However, with only 2,500 drivers in the system, we see the dispatch rates go down to about 95%. Such results suggest that the appropriate number of drivers for this system might be close to 3,000 drivers, located near-optimally using the iterative allocation scheme. We also observe that in all cases, even with 2,500 drivers, not all drivers are being used. This is primary the result of using a two-phase driver allocation scheme, where bid drivers are allocated independently from extra-board drivers. Since extra-boards are quite flexible, when bid drivers and extra-boards are included simultaneously it turns out that even fewer drivers are needed, since the extra-board drivers can perform bid work when no bid drivers are available. While such an observation may point to developing a methodology that allocates bid drivers and extra-board drivers simultaneously, this is actually very difficult to do due to the "poaching" bias that we describe earlier in Section 4.2.

4.3.3 Experiment 3: Allowing Only Extra-board Drivers in the System

LTL carriers realize that union rules add costs to their operations, but quantifying these additional costs is difficult. In this final experiment, we use the driver management technology and the driver allocation scheme to develop initial insights regarding the impact of union rules on the number of drivers required in the system.

To do so, we use the phase II driver allocation scheme of Algorithm 4.1 to domicile 2,000 extra-board drivers for load set instances W1, W2, and W3. Then, we develop a final best allocation using the optimization models given in Section 4.2.4. Finally, we develop driver schedules for each of the load set instances using the driver scheduling technology to generate the final performance measure results of operating a system with 2,000 well-located extra-board drivers. Table 4.4 summarizes the results.

We observe that a 97% dispatch rate can be achieved with only 2,000 extra-board drivers well-located in the system, whereas 3,000 are needed to achieve a similar dispatch rate if $\rho = \frac{2}{3}$ of the drivers are designated as bid drivers. That is, 50% more drivers are required to achieve the same level of service. Observe that, not surprisingly, there is a significant

Table 4.4: Performance statistics using the iterative allocation scheme with only extra-board drivers in the system.

Date	#Drivers Assigned	%Loads Dispatched	%Empty	%Foreign Beds	Average Drive Time	Average Duty Time	Tours per driver
W1	2000	97.67	5.96	75.74	9.28	10.54	5.36
W2	1994	96.11	6.70	76.98	9.14	10.43	5.27
W3	1996	97.73	6.13	75.65	9.29	10.56	5.21

increase (20%) in the number of foreign beds when there are only extra-board drivers. The driver management technology is exploiting the additional flexibility offered by extra-board drivers.

CHAPTER V

FUNDAMENTAL ANALYSIS OF DRIVER MANAGEMENT PROBLEMS

5.1 Introduction

One of the questions pertinent to both driver management and fleet sizing is the number of drivers required to dispatch a given set of loads. The number of drivers is a good proxy to determine the utilization of drivers: fewer drivers implies better utilization. Moreover, LTL carriers prefer to recruit as few drivers as possible to minimize the fixed costs associated with having drivers on the payroll. However, determining the minimum number of drivers is complicated because of the complex constraints governing the drivers. In this chapter, we aim to provide basic analysis and insightful results regarding this problem under the complicating U.S. DOT hours-of-service requirement. These requirements transform the drivers into renewable resources and scheduling drivers becomes challenging. To begin with, it is useful to pose the simplest setting that allows investigation of this problem.

5.2 Two-Terminal Driver Assignment Problem (2DAP)

It is challenging to provide theoretical results in terms of the complex structure of real-life LTL operations and terminal network. Therefore, we make two simplifying assumptions:

- The network consists of only two terminals; and
- The loads have a fixed dispatch time, instead of a time window.

We define the Two-Terminal Driver Assignment Problem (2DAP) as: Given a set $\mathcal{L} = \{1 \dots n\}$ of loads over a fixed time horizon between two terminals, A and B, determine the minimum number drivers to locate at each terminal, such that all dispatches are feasibly served by a driver. Let m_A and m_B be the driver pool m_B be the driver pool sizes at A and B respectively; then we want to minimize the sum, $m_{opt} = m_A + m_B$.

Suppose, for both directions, the drive time and run time are known with certainty and equal to tt (*i.e.*, $\textsc{time}(A, B) = tt$, $\textsc{time}(B, A) = tt$, $\textsc{rtime}(A, B) = tt$ and $\textsc{rtime}(A, B) = tt$). Let $\textsc{orig}(\ell)$ and $\textsc{dest}(\ell)$ denote the origin and destination of load ℓ respectively. Precise dispatch time of each load $\ell \in \mathcal{L}$ implies that $\textsc{ready}(\ell) = \textsc{cut}(\ell)$. Therefore, the dispatch time will be denoted by $r_\ell \in \mathbb{Z}^+$, replacing our notation of from previous chapter for loads with flexible time windows. Without loss of generality, we can assume that loads are sorted in non-decreasing order of dispatch times (*i.e.*, $r_i \leq r_j$ if $i < j$). Empty dispatches from A to B, or from B to A are available at any time.

In practice, as explained earlier in this dissertation, drivers are constrained by work rules that limit the hours that can be driven between rests, and the total time spent on duty between rests. Rests are mandated to have a minimum duration. To study the impact of such restrictions, we analyze variants with additional constraints.

Assumption 5.1 (Drive Hours Restricted (V))**.** *Each driver may drive no more than τ_V hours before resting for a minimum of τ_R hours.*

Assumption 5.2 (Duty Hours Restricted (U))**.** *Each driver may be on duty for no more than τ_U hours before resting for a minimum of τ_R hours. A driver begins a duty at the time of his first dispatch off rest.*

In the following sections, we will consider four different variants:

1. 2DAP: In this version, drivers are not restricted in terms of drive time or duty time requirements. A driver's schedule will be a sequence $s_1, s_2, \ldots$ of moves, with a load or empty. There is only one feasibility requirement for his schedule: the arrival time of each move is before the dispatch time of the succeeding one (*i.e.*, $d_{s_i} + tt \leq d_{s_{i+1}}$).

2. 2DAP(V): In 2DAP(V), drivers are restricted in terms of drive time. A driver's schedule will be a sequence of moves with rests in between. In addition to the feasibility requirement from 2DAP, the sum of the drive time of consecutive moves between rests should not be more than τ_V hours. Once a driver accumulates a total driving time of τ_V hours, he must rest a minimum of τ_R hours.

3. 2DAP(U): In 2DAP(U), drivers are restricted in terms of duty time. A driver's schedule will be a sequence of moves with rests in between. In addition to the feasibility requirement of 2DAP, the driver cannot *work* more than τ_U hours between rest. The work of a driver includes the waiting time in between two moves. Once a driver accumulates a total of τ_U duty hours, he must rest a minimum of τ_R hours.

4. 2DAP(U,V): In this version, all the requirements of 2DAP(U) and 2DAP(V) are present. A driver cannot drive more than τ_V hours and cannot work more than τ_U hours. Once he runs out of either duty hours or drive hours, he must rest a minimum of τ_R hours.

Observation 5.3. *Every instance of each 2DAP version has a feasible solution, where* $m_{opt} = m_A + m_B \leq |\mathcal{L}| = n$.

The remainder of this chapter is organized as follows: In Section 5.3, we analyze 2DAP, the problem with no work rule constraints on the driver schedules and provide two polynomial-time algorithms that solve it to optimality. In Section 5.4, we analyze the problem 2DAP(V), the variant with limits on drive time, and provide lower bounds, heuristics, and an integer programming formulation. In Section 5.5, we focus on a special case of 2DAP(V), where the driver is not allowed to move more than twice between rests. Section 5.6 presents a computational study on the performance of lower bounds and heuristics related to 2DAP and 2DAP(V). Finally, in Section 5.7, we propose solution procedures for 2DAP(U) and 2DAP(U,V), the two versions where the drivers are limited in terms of duty time.

5.3 Solving 2DAP

We define 2DAP to be the version without any work rule constraints on driver schedules. Let us start with an observation regarding driver schedules.

Observation 5.4. *As mentioned earlier, the schedule of a driver will be an ordered sequence of loads and empty moves in 2DAP. The waiting time in between moves is not important and there is no incentive for the driver to delay an empty move. If there is an empty move*

in the driver's schedule, we can assume that the empty move is executed immediately after the preceding load. That is, the dispatch time will be $r_\ell + tt$ for an empty move immediately following load ℓ.

5.3.1 Lower Bounds

Occupation Bound

We can generate a first lower bound on the total number of drivers required by a simple *occupation* rule. Each load $\ell \in \mathcal{L}$ will occupy a driver during the interval $[r_\ell, r_\ell + tt)$. Whenever intervals for different loads overlap each other, an individual driver is needed for each load. The maximum number of overlapping loads then is a lower bound on the number of drivers. More formally, let

$$o_t^\ell = \begin{cases} 1 & t \in [r_\ell, r_\ell + tt) \\ 0 & \text{otherwise} \end{cases}$$

Then, a bound on the drivers needed at time t, $\underline{\mathrm{m}}_t$ is now

$$\underline{\mathrm{m}}_t = \sum_{\ell \in \mathcal{L}} o_t^\ell$$

and a lower bound for the system is then

$$\underline{\mathrm{m}}^O = \max_t \underline{\mathrm{m}}_t$$

This method for calculating the occupation bound requires calculating $\underline{\mathrm{m}}_t$ for every possible t, which is computationally cumbersome and unnecessary. Another way to calculate this bound is to simply sum the number of dispatches from $\textsc{orig}(\ell)$ in the interval $[r_\ell, r_\ell + tt)$ and arrivals to $\textsc{orig}(\ell)$ in the interval $[r_\ell + tt, r_\ell + 2tt)$, for each load $\ell \in \mathcal{L}$. Let,

$$\begin{aligned} O_\ell = &\{\ell' \in \mathcal{L} \mid orig(\ell') = orig(\ell), r_{\ell'} \in [r_\ell, r_\ell + tt)\} \cup \\ &\{\ell' \in \mathcal{L} \mid orig(\ell') \neq orig(\ell), r_{\ell'} \in [r_\ell + tt, r_\ell + 2tt)\},\ \forall \ell \in \mathcal{L} \end{aligned}$$

Note that $|O_\ell| = \underline{\mathrm{m}}_{r_\ell + tt}$. Then, occupation bound is

$$\underline{\mathrm{m}}^O = \max_{\ell \in \mathcal{L}} |O_\ell| .$$

Using this approach, the occupation bound can be calculated in $\mathcal{O}(n^2)$ operations.

Cycle Bound

It should be noted that the occupation bound does not account for the location of the drivers when they finish a load. Suppose, for example, that loads are dispatched from A to B every tt hours, and no loads return. Since there is never more than one driver occupied by a load, the occupation bound $\underline{\mathrm{m}}^O = 1$. It is clear, however, that two drivers are needed in this system, where a driver is returning empty from B to A every tt hours.

With this in mind, we can improve the occupation bound by using the concept of driver *cycles*. The idea is essentially as follows: when a driver is dispatched, say from A to B, with load ℓ at time r_ℓ, that driver cannot be dispatched again from A to B until at least time $r_\ell + 2tt$, since the driver must travel from A to B loaded and then from B to A, with a load or empty. Thus, if we count the total number of loads dispatched from A to B during the interval $[r_\ell, r_\ell + 2tt)$, we know we need unique drivers for each of them. Repeating this process for each load generates another bound. More formally, let

$$C_\ell = \{\ell' \in \mathcal{L} \mid orig(\ell') = orig(\ell), r_{\ell'} \in [r_\ell, r_\ell + 2tt)\},\ \forall \ell \in \mathcal{L}$$

Then, the cycle bound is

$$\underline{\mathrm{m}}^C = \max_{\ell \in \mathcal{L}} |C_\ell| .$$

Although cycle bound captures the driver location information better, it is not necessarily a better bound than the occupation bound as demonstrated by Example 5.5.

Example 5.5. *2 loads.* $r_1 = 0$, $r_2 = 0$, ORIG*(1)* $= A$, ORIG*(2)* $= B$.
In this instance, $\underline{m}^O = 2$ *and* $\underline{m}^C = 1$.

Improved Cycle Bound

The drawback of the cycle bound is that it only considers dispatches outbound from the same terminal as the load ℓ under consideration, and as in Example 5.5, it might not be tight. It can be improved by overlapping the dispatches from and arrivals to ORIG(ℓ) that happen in the interval $[r_\ell, r_\ell + 2tt)$.

Suppose we are considering the interval $[r_\ell, r_\ell + 2tt)$, and that ℓ is a dispatch from A to B. Let $\ell_{\max}$ be the latest dispatch from A to B where $r_{\ell_{\max}} \in [r_\ell, r_\ell + 2tt)$. Now, suppose that we have at least one dispatch ℓ' from B to A that arrives at A in the interval $(r_{\ell_{\max}}, r_\ell + 2tt)$. Each of these dispatches needs a driver different from those counted using the cycle bound procedure for ℓ because every dispatch from A to B that is counted must arrive after $r_{\ell'}$.

We can generalize this idea to count for, not only arrivals after $r_{\ell_{\max}}$, but for all arrivals in $[r_\ell, r_\ell + 2tt)$. Consider all loads arriving to the origin terminal ℓ after r_ℓ but before $r_\ell + 2tt$ (*i.e.*, in $(r_\ell, r_\ell + 2tt)$), and suppose the first such load arrives at time t_a. Starting at time t_a, we can perform a simple count to determine how many additional drivers are required by inbound loads. In this *improvement step*, each time an arriving load occurs, increment the count and each time a departing load occurs, decrement the count (but never below 0). Add this number to the cycle bound to generate the improved cycle bound, $\underline{\mathrm{m}}^{IC}$.

We can combine the cycle bound calculation and the improvement step. As mentioned earlier in the construction of the cycle bound, any departure from $\textsc{orig}(\ell)$ in the interval $[r_\ell, r_\ell + 2tt)$ will require a separate driver. This is also true for arrivals to $\textsc{orig}(\ell)$ in the interval. We can simply "match" the arrivals to departures, by iterating over the loads sequentially in time. Matched loads can be covered by the same driver, however loads that are not matched will require separate driver. Let

$$\begin{aligned}\mathcal{L}_\ell = &\{\ell' \in \mathcal{L} \mid \textsc{orig}(\ell') = \textsc{orig}(\ell),\ r_{\ell'} \in [r_\ell, r_\ell + 2tt)\} \cup \\ &\{\ell' \in \mathcal{L} \mid \textsc{orig}(\ell') \neq \textsc{orig}(\ell),\ r_{\ell'} \in [r_\ell - tt, r_\ell + tt)\}.\end{aligned}$$

For each $\ell' \in \mathcal{L}_\ell$

$$\textsc{type}(\ell') = \begin{cases} \textsc{dep} & \textsc{orig}(\ell) = \textsc{orig}(\ell') \\ \textsc{arr} & \text{otherwise} \end{cases}$$

and

$$T_{ad}(\ell') = \begin{cases} r_{\ell'} & \text{ORIG}(\ell) = \text{ORIG}(\ell') \\ r_{\ell'} + tt & \text{otherwise} \end{cases}$$

For each load $\ell' \in \mathcal{L}_\ell$, $\text{TYPE}(\ell')$ indicates if the load is arriving or departing, and $T_{ad}(\ell')$ denotes arrival time or dispatch time depending on the type of load. Then, the method for calculating the improved cycle bound for load ℓ, denoted $\underline{\text{m}}_\ell^{IC}$, is given in Algorithm 5.1.

Algorithm 5.1 Calculating improved cycle bound

```
Sort ℓ' ∈ L_ℓ in ascending T_ad. Break ties in favor of loads with TYPE(ℓ') = ARR
m_ℓ^IC ← 0
a_c ← 0
for all (Loads ℓ' in the sorted list) do
  if (TYPE(ℓ') = DEP) then
    if (a_c = 0) then
      m_ℓ^IC ← m_ℓ^IC + 1
    else
      a_c ← a_c − 1
    end if
  else
    m_ℓ^IC ← m_ℓ^IC + 1
    a_c ← a_c + 1
  end if
end for
return m_ℓ^IC
```

Then, the improved cycle bound is

$$\underline{\text{m}}^{IC} = \max_{\ell \in \mathcal{L}} \underline{\text{m}}_\ell^{IC}.$$

By construction, $\underline{\text{m}}^{IC} \geq \underline{\text{m}}^{C}$. The improved cycle bound is also better than the occupation bound: when calculating the bounds for a load ℓ, all dispatches from the origin that are considered in the occupation bound (*i.e.*, all dispatches in the interval $[r_\ell, r_\ell + tt)$) are already accounted for within the original cycle bound definition. The arrivals in the interval $[r_\ell + tt, r_\ell + 2tt)$ are either counted as a part of original cycle bound definition if there is a load dispatched after an arrival, or counted directly by the improvement step. Example 5.6

shows that improved cycle bound can be strictly better than both the occupation bound and the cycle bound.

Example 5.6. *6 loads,* $tt = 5$*,*

ℓ	r_ℓ	ORIG(ℓ)
1	0	A
2	3	B
3	4	B
4	6	A
5	9	A
6	12	B

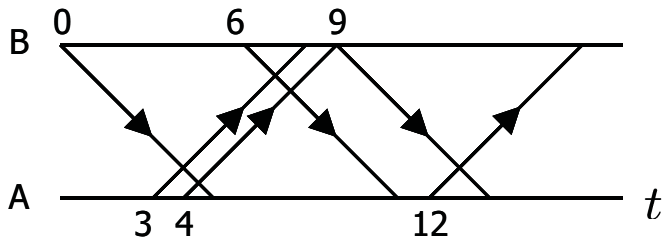

Figure 5.1: Dispatch details for Example 5.6

In this instance, $\underline{m}^C = 3$*,* $\underline{m}^O = 3$*, and* $\underline{m}^{IC} = 4$*.*

5.3.2 Optimal Solution Procedure

In this section, we will provide two polynomial-time algorithms that solve 2DAP to optimality. The first one uses the improved cycle bound construction as basis. The second one is a minimum cost flow circulation model.

Without loss of generality, let us assume that the origin of the load with the earliest dispatch time is A. Then, for each load $\ell \in \mathcal{L}$ define,

$$\text{TYPE}(\ell) = \begin{cases} \text{DEP} & \text{ORIG}(\ell) = A \\ \text{ARR} & \text{otherwise} \end{cases}$$

and

$$t_{ad}(\ell) = \begin{cases} r_\ell & \text{ORIG}(\ell) = A \\ r_\ell + tt & \text{otherwise} \end{cases}$$

In the first optimal solution procedure, set of drivers is denoted by $\mathcal{D}$. Each driver d has two attributes: LOCATION(d) and READY(d) denoting the location and the ready time

of the driver respectively. $\mathcal{S}_d$ represents the set of loads that are assigned to driver d. The algorithm iterates over loads in ascending order of t_{ad}. Each load ℓ is assigned to a driver in $\mathcal{D}$ that can cover the load feasibly. Precedence is given to drivers that are at the origin of the load. If there are multiple drivers, the load is assigned to the one with the maximum ready time. By assigning the driver with maximum ready time, we maintain the most flexibility in case we need to send a driver empty to the other terminal. Algorithm 5.2 presents a pseudo-code for the optimal procedure to solve 2DAP.

Algorithm 5.2 Solving 2DAP

Let $\underline{\mathcal{L}}$ be list of loads sorted in ascending order of t_{ad}. Break ties in favor of loads with $\text{TYPE}(\ell) = \text{ARR}$
$\mathcal{D} \leftarrow \emptyset$
for all $(\ell \in \underline{\mathcal{L}})$ **do**
  $\mathcal{D}' \leftarrow \{d \in \mathcal{D} \mid \text{READY}(d) \leq r_\ell,\ \text{LOCATION}(d) = \text{ORIG}(\ell)\}$
  if $(\mathcal{D}' \neq \emptyset)$ **then**
    $d \leftarrow \text{argmax}_{d' \in \mathcal{D}'} \text{READY}(d')$
  else
    $\mathcal{D}' \leftarrow \{d \in \mathcal{D} \mid \text{READY}(d) \leq r_\ell - tt,\ \text{LOCATION}(d) \neq \text{ORIG}(\ell)\}$
    if $(\mathcal{D}' \neq \emptyset)$ **then**
      $d \leftarrow \text{argmax}_{d' \in \mathcal{D}'} \text{READY}(d')$
    else
      Let d be a new driver
      $\mathcal{S}_d \leftarrow \emptyset$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{d\}$
    end if
  end if
  $\mathcal{S}_d \leftarrow \mathcal{S}_d \cup \{\ell\}$
  $\text{READY}(d) \leftarrow r_\ell + tt$
  $\text{LOCATION}(d) \leftarrow \text{DEST}(\ell)$
end for

Theorem 5.7. *Algorithm 5.2 uses at most $\underline{m}^{IC}$ drivers.*

Proof: Assume that, at some iteration of the algorithm, $|\mathcal{D}| = \underline{\text{m}}^{IC}$, and a new driver is required to dispatch load ℓ. Note that all drivers d located at A will have $\text{READY}(d) \leq r_\ell$, because Algorithm 5.2 iterates over the arrivals to A in order of arrival times. By similar reasoning, there cannot be any drivers at B with ready time later than $r_\ell + tt$.

Case 1: Assume $\text{ORIG}(\ell) = A$. There cannot be any drivers located at A, or any drivers located at B with ready time before $r_\ell - tt$. Otherwise, such a driver would be able to

cover ℓ (drivers at B need to move empty first). Therefore, all drivers are at B with ready times in the interval $(r_\ell - tt, r_\ell + tt]$. Let $\mathcal{L}'$ denote the set of loads that the drivers covered last (See Figure 5.2). Observe that $|\mathcal{L}'| = \underline{\mathrm{m}}^{IC}$, and each load in $\mathcal{L}'$ has dispatch time in the interval $(r_\ell - 2tt, r_\ell]$. Let $\ell' \in \mathcal{L}'$ be the load with the earliest dispatch time. By definition, $\mathcal{L}' \subset C_{\ell'}$. Since $r_\ell - r_{\ell'} < 2tt$, $\ell \in C_{\ell'}$. Then, $|C_{\ell'}| \geq |\mathcal{L}' \cup \{\ell\}| = \underline{\mathrm{m}}^{IC} + 1 > \underline{\mathrm{m}}^{IC}$, contradicting $\underline{\mathrm{m}}^{C} \leq \underline{\mathrm{m}}^{IC}$.

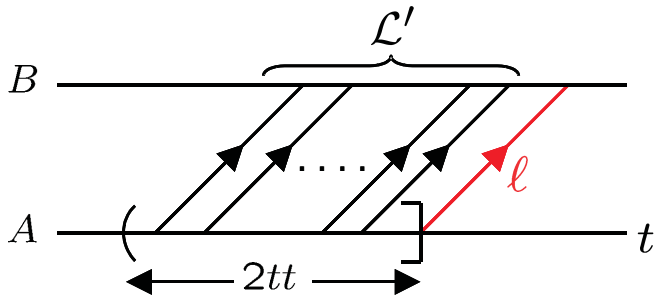

Figure 5.2: 2DAP - Proof Case 1

Case 2: Assume $\textsc{orig}(\ell) = B$. Since there are no drivers to cover load ℓ, drivers at B have ready time later than r_ℓ, and drivers at A have ready time later than $r_\ell - tt$. If we assume there are no drivers at B, we can construct a contradiction similar to the previous case: Let $\mathcal{L}'$ denote the set of loads that the drivers covered last. Observe that $|\mathcal{L}'| = \underline{\mathrm{m}}^{IC}$, and each load in $\mathcal{L}'$ has dispatch time in the interval $(r_\ell - 2tt, r_\ell]$. Let $\ell' \in \mathcal{L}'$ be the load with the earliest dispatch time. By definition, $\mathcal{L}' \subset C_{\ell'}$. Since $r_\ell - r_{\ell'} < 2tt$, $\ell \in C_{\ell'}$. Then, $|C_{\ell'}| \geq |\mathcal{L}' \cup \{\ell\}| = \underline{\mathrm{m}}^{IC} + 1 > \underline{\mathrm{m}}^{IC}$, contradicting $\underline{\mathrm{m}}^{C} \leq \underline{\mathrm{m}}^{IC}$.

Next, assume that there are drivers at B. Let $\mathcal{L}''$ and $\mathcal{L}'$ denote the set of loads that the drivers located at A and B covered last. If for any $\ell' \in \mathcal{L}'$, there is a load in $\hat{\ell} \in \mathcal{L}''$ such that $r_{\hat{\ell}} + tt \leq r_{\ell'}$, then there is a load ℓ'' that the driver has covered preceding ℓ'. If there was no such load or the move was an empty, then the driver that covered load $\hat{\ell}$ would be assigned to load ℓ' by the tie-breaker rule. Add such load ℓ'' to $\mathcal{L}''$. Note that each ℓ'' is dispatched from B and has a dispatch time in the interval $(r_\ell - 2tt, r_\ell]$ (See Figure 5.3).

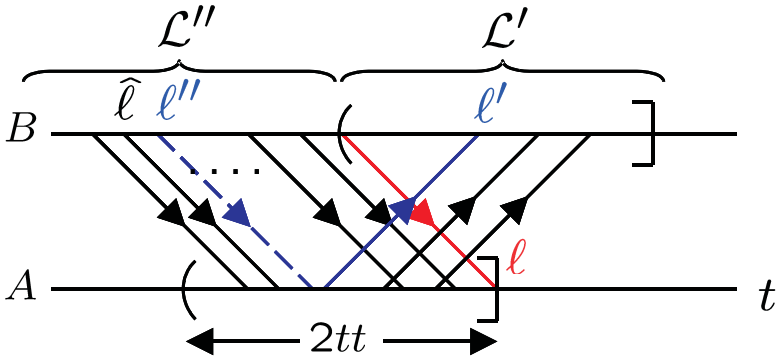

Figure 5.3: 2DAP - Proof Case 2

Case 2a: Assume that there are no loads $\ell' \in \mathcal{L}'$ such that there is no load in $\hat{\ell} \in \mathcal{L}''$ with $r_{\hat{\ell}} + tt \leq r_{\ell'}$. Then, $\mathcal{L}''$ consists of $|\mathcal{D}| = \underline{\mathrm{m}}^{IC}$ loads, all dispatched from B in the interval $(r_\ell - 2tt, r_\ell]$. Let $\bar{\ell}$ the one with the earliest dispatch time. Then, $\mathcal{L}'' \subset C_{\bar{\ell}}$. Since $r_\ell - r_{\bar{\ell}} < 2tt$, $\ell \in C_{\bar{\ell}}$. Then, $|C_{\bar{\ell}}| \geq |\mathcal{L}'' \cup \{\ell\}| = \underline{\mathrm{m}}^{IC} + 1 > \underline{\mathrm{m}}^{IC}$, contradicting $\underline{\mathrm{m}}^{C} \leq \underline{\mathrm{m}}^{IC}$.

Case 2b: Assume that there are loads $\ell' \in \mathcal{L}'$ such that there is no load in $\hat{\ell} \in \mathcal{L}''$ with $r_{\hat{\ell}} + tt \leq r_{\ell'}$. Let $\overline{\mathcal{L}}$ be the set of such loads (See Figure 5.4). Note that all loads in $\overline{\mathcal{L}}$ are dispatched from A, have a dispatch time in $(r_\ell - tt, r_\ell + tt]$, and their dispatch times are earlier than arrival times of the loads in $\mathcal{L}''$. Let $\bar{\ell} \in \overline{\mathcal{L}}$ be the one with the earliest dispatch time. Then, $\overline{\mathcal{L}} \subset IC_{\bar{\ell}}$. By construction, each load $\ell'' \in \mathcal{L}''$ is either in $IC_{\bar{\ell}}$ or there is a succeeding ℓ'' covered by the same driver in $IC_{\bar{\ell}}$. Since $r_\ell + tt - r_{\bar{\ell}} < 2tt$, either $\ell \in IC_{\bar{\ell}}$, or there is a dispatch from A coupled with ℓ in $IC_{\bar{\ell}}$. Then $|IC_{\bar{\ell}}| \geq |\overline{\mathcal{L}}| + |\mathcal{L}''| + |\{\ell\}| = \underline{\mathrm{m}}^{IC} + 1 > \underline{\mathrm{m}}^{IC}$, contradicting $|IC_{\bar{\ell}}| \leq \underline{\mathrm{m}}^{IC}$.

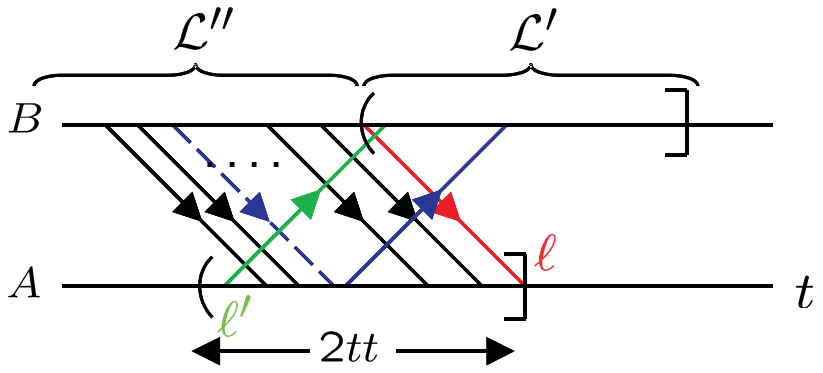

Figure 5.4: 2DAP - Proof Case 2b

■

Theorem 5.8. *Algorithm 5.2 solves 2DAP optimally in $\mathcal{O}(n^2)$ time.*

Proof: By Theorem 5.7, Algorithm 5.2 uses at most $\underline{m}^{IC}$, which is a lower bound on the number of drivers required. Therefore, the algorithm produces optimal solution. There are n iterations of the algorithm, one for each load. At each iteration, the load is checked against drivers in set $\mathcal{D}$ for feasibility. Since $|\mathcal{D}| \leq n$, the algorithm complexity is $\mathcal{O}(n^2)$. ■

Next, we will present another solution procedure based on a minimum cost circulation model: Consider the solution of the minimum cost $s - t$ flow on the following network $G = (\mathcal{N}, \mathcal{A})$ with $\mathcal{N} = \{p_j, j = 1 \ldots n\} \cup \{s, t\}$. Each p_j node corresponds to a load j.

The arcs and the bounds on the arcs are as follows (assume a lower bound of 0 and no upper bound on an arc if not mentioned explicitly) :

- There is an arc from s to each $p_j, j = 1 \ldots n$ with a cost of 1 and an upper bound of 1.
- All $p_j, j = 1 \ldots n$ has a lower bound of 1 (A lower bound on a node p can be converted to a lower bound on an arc by replacing p by two nodes, p' and p'', and adding an arc from p' to p'' with the lower bound. All inbound arcs to p are connected to p' and outbound arcs start at p'').
- There is an arc of cost 0 between p_i and p_j, if:

1. $r_i + tt \leq r_j$ and $\text{DEST}(i) = \text{ORIG}(j)$
2. $r_i + 2tt \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$.

In both cases, a flow on the arc corresponds to a driver covering first load i and then j. In the second case, there is an empty move between the two loads.

- All p_j nodes are connected to t with a cost of 0.
- t is connected to s.

Proposition 5.9. *By construction, the minimum cost circulation on G gives the minimum number of drivers needed to cover the loads and each path of unit flow from s to t will correspond to driver duty.*

We can extend the circulation model to the case where there are more than two terminals. The requirements on the arcs between p_i and p_j is the same if $\text{DEST}(i) = \text{ORIG}(j)$. If destination of i and origin of j are different, then the travel time between the two terminals is required to construct the network. The number of nodes is only dependent on the number of loads, not on the network size. Therefore, the solution procedure is still polynomial.

Even though we can relax the simplifying assumption of having only two terminals, we cannot relax the fixed dispatch time requirement of loads. Once the loads have a dispatch window, the transitive property of loads sequentially being covered is no longer valid. That is, if loads i and j can be covered feasibly by a single driver and loads j and k can be covered feasibly by a single driver does not imply that loads i, j and k can be covered by a single driver. Therefore, the above model will not be able to solve the problem with time windows.

5.4 Solving 2DAP with Drive Time Restrictions (2DAP(V))

In this section, we consider 2DAP(V), the case where each driver may drive no more than τ_V hours before resting for a minimum of τ_R hours. Let $k = \lfloor \tau_V / tt \rfloor$ denote the maximum number of moves, with a load or empty, a driver can do between any two rests.

Observation 5.10. *As mentioned earlier, the schedule of a driver will be a sequence of loads and empty moves with rests in between. In 2DAP(V), the cardinality of moves between rests*

is important, not how long a driver waits between moves. If there is an empty move in the schedule of a driver, we can assume that the empty move is executed immediately after the preceding load. That is, the dispatch time will be $r_\ell + tt$ for an empty move immediately following load ℓ.

Observation 5.11. *Without loss of generality we can assume that $k \leq 2\,n$, otherwise the constraint that a driver has to rest will never be binding and the problem will be the same as 2DAP.*

The critical case for the validity of the above observation is when only one driver covers all the loads. In such a case, if $k < n$, the driver needs to rest at least once because there are more loads than he can cover without resting. It is possible that the driver may move empty. Since we have only 2 terminals, he does not need to move empty more than once between two loads. Assuming the worst case, *i.e.*, the driver moves empty before each load, the driver will move $2\,n$ times (n times with a load and n times empty). Thus, if $k > 2\,n$, the driver will never need to go to rest.

Observation 5.12. *2DAP is a lower bound for 2DAP(V), for any value of τ_R and τ_V.*

5.4.1 Polynomially Solvable Cases

Observation 5.13. *We can solve 2DAP(V) where $k = 1$ in polynomial time.*

If $\tau_V/2 < tt \leq \tau_V$, then a driver can do at most one move (with a load or empty) between two rests. The solution to this problem can be obtained by a minimum cost circulation formulation similar to the one in Section 5.3.2 with minor modifications on the arcs. There is a zero-cost arc between p_i and p_j if:

1. $r_i + tt + \tau_R \leq r_j$ and $\text{DEST}(i) = \text{ORIG}(j)$
2. $r_i + 2tt + 2\tau_R \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$.

The arcs from s to p nodes and from p nodes to t are same as in Section 5.3.2.

Observation 5.14. *We can solve 2DAP(V) where $k = 2$ and all loads originate at the same terminal in polynomial time.*

Without loss of generality, let us assume that all loads originate at A. The network in Section 5.3.2 is still valid. Since all loads originate at A, a driver has to return to A with an empty to cover another load and this is the only feasible option. Therefore, there is an arc between p_i and p_j if $r_i + 2tt + \tau_R \leq r_j$.

5.4.2 Integer Programming Formulation

We can formulate 2DAP(V) as an integer programming formulation. Consider the following circulation problem on $G = (\mathcal{N}, \mathcal{A})$ with $\mathcal{N} = \{p_{i,j}, i = 1 \dots n, j = 1 \dots k\} \cup \{s, t\}$: All nodes are transshipment nodes, *i.e.*, $b(s) = 0,\ b(t) = 0,\ b(p_{i,j}) = 0$. A node $p_{i,j}$ represents that load i is covered as the j^{th} load after a rest. For example, $p_{i,1}$ represents load i to be covered right after a rest, $p_{i,2}$ represent load i to be covered as the second move after a rest etc.

Assume a cost of 0, a lower bound of 0 and an upper bound of ∞ on an arc if not mentioned explicitly. There is an arc from s to $p_{i,1}, i = 1 \dots n$ with an upper bound of 1. All $p_{i,j}, i = 1 \dots n,\ j = 1 \dots k$ are connected to t. There are five conditions on loads that allow arcs among $p_{i,j}$ and these arcs represent feasible assignments to a driver.

1. There is an arc between $p_{i,h}$ and $p_{j,h+1},\ h = 1 \dots k-1$ if $r_i + tt \leq r_j$ and $\text{DEST}(i) = \text{ORIG}(j)$: This represents the case where load j can be executed immediately after load i.

2. There is an arc between $p_{i,h}$ and $p_{j,h+2},\ h = 1 \dots k-2$ if $r_i + 2tt \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$: This arc represents the case where load i and j are originating from the same terminal and load i is followed by an empty move and then load j.

3. There is an arc between $p_{i,h}$ and $p_{j,1}, h = 1 \dots k$ if $r_i + tt + \tau_R \leq r_j$ and $\text{DEST}(i) = \text{ORIG}(j)$: This arc represents the case where there is a rest after load i and load j comes after the rest.

4. There is an arc between $p_{i,h}$ and $p_{j,1}, h = 1 \dots k-1$ if $r_i + 2tt + \tau_R \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$: This arc represents the case where i and j are originating from the same

terminal and there is an empty move and a rest after load i and load j comes after the rest.

5. There is an arc between $p_{i,k}$ and $p_{j,2}$ if $r_i + 2tt + \tau_R \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$: This arc represents the case where load i is the last load before a rest. After the rest, there is an empty move and load j.

There is an arc from t to s, the feedback arc, with a cost of 1 and no upper bound.

To make sure that all loads are covered by a driver we need an additional set of constraints called *cover constraints*. Let x_{ij}^{qr} represent the flow variable on arc from node $p_{i,j}$ to $p_{q,r}$. Then,

$$\sum_{j=1\ldots k} \sum_{q \geq i} \Big(\sum_{r>j} x_{ij}^{qr} + x_{ij}^{q1} \Big) = 1, \ \forall i = 1 \ldots n$$

Figure 5.5 gives a graphical representation of the integer programming model for 2DAP(V).

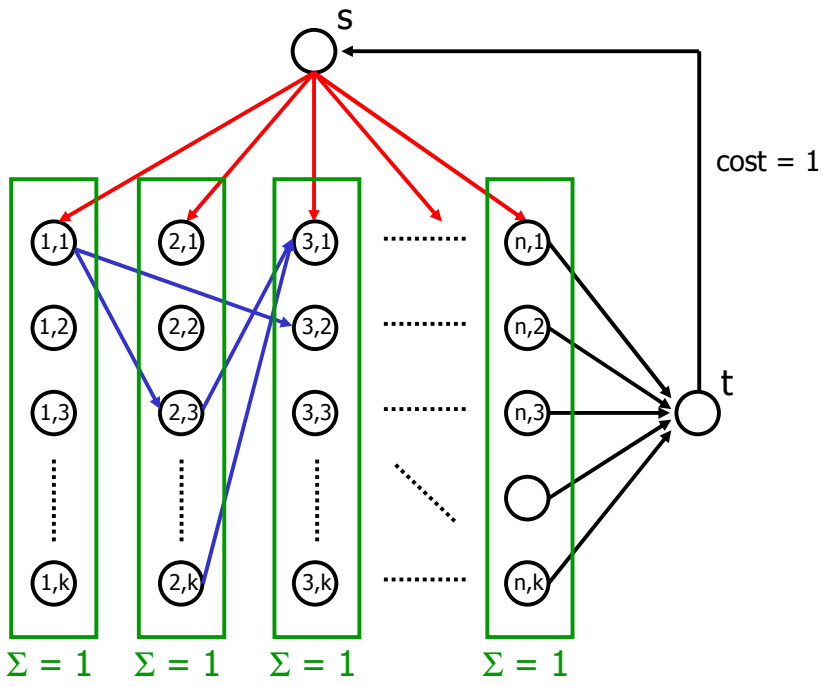

Figure 5.5: Integer Programming Model for 2DAP(V)

With the addition of cover constraints, the optimal solution can have fractional flow.

Therefore, we need to require the flow variables to be binary making the above formulation an integer program.

Proposition 5.15. *Minimizing the total flow in the above formulation results with minimum number of drivers for 2DAP(V), and by construction, each unit flow from s to t corresponds to a driver schedule.*

Note that the number of nodes in the network is $\mathcal{O}(n^2)$ because there are $k \times n + 2$ nodes in the network and by Observation 5.11, $k \leq 2\,n$.

5.4.3 Heuristics

Longest Path Heuristic

The Longest Path Heuristic (LPH) iteratively finds a driver duty that includes the maximum number of loads in $\mathcal{U}$, the set of undispatched loads, and assigns it to a driver. Initially, $\mathcal{U}$ is equal to the set of all loads. Let m_l represent the number of drivers used in the algorithm. Algorithm 5.3 gives the pseudo-code of LPH.

Algorithm 5.3 Pseudo-Code of Longest Path Heuristic

$\mathcal{U} \leftarrow \mathcal{L}$
$m_l \leftarrow 0$.
repeat
  Find the longest sequence $\mathcal{S}$ of loads a driver can cover with loads from $\mathcal{U}$.
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}$
  $m_l \leftarrow m_l + 1$.
until $(\mathcal{U} = \emptyset)$
return m_l

The number of iterations in LPH is clearly bounded by n. Next, we provide a shortest path algorithm for finding the longest sequence of loads that can be feasibly covered by a single driver, making LPH a polynomial time heuristic.

Consider the following network on $G = (\mathcal{N}, \mathcal{A})$ with $\mathcal{N} = \{p_{i,j}, i = 1 \ldots n, j = 1 \ldots k\} \cup \{s, t\}$: Node $p_{i,j}$ represents load i being covered as the j^{th} move after a rest. For example, $p_{i,1}$ represents load i is covered right after a rest, $p_{i,2}$ represents load i is covered as the second move after a rest etc. There is an arc from s to $p_{i,1}, i = 1 \ldots n$. All $p_{i,j}, i = 1 \ldots n,\ j = 1 \ldots k$

are connected to t. There are five conditions on loads that allow arcs among $p_{i,j}$ which represent feasible assignments to a driver.

- There is an arc between $p_{i,h}$ and $p_{j,h+1},\ h = 1 \ldots k-1$ if $r_i + tt \leq r_j$ and $\text{DEST}(i) = \text{ORIG}(j)$: This arc represents the case where load j can be executed immediately after load i.

- There is an arc between $p_{i,h}$ and $p_{j,h+2},\ h = 1 \ldots k-2$ if $r_i + 2tt \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$: This arc represents the case where load i and j are originating from the same terminal and load i is followed by an empty and load j.

- There is an arc between $p_{i,h}$ and $p_{j,1}, h = 1 \ldots k$ if $r_i + tt + \tau_R \leq r_j$ and $\text{DEST}(i) = \text{ORIG}(j)$: This arc represents the case where there is a rest after load i and load j comes after the rest.

- There is an arc between $p_{i,h}$ and $p_{j,1}, h = 1 \ldots k-1$ if $r_i + 2tt + \tau_R \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$: This arc represents the case where i and j are originating from the same terminal and there is an empty move and a rest after load i and load j comes after the rest.

- There is an arc between $p_{i,k}$ and $p_{j,2}$ if $r_i + 2tt + \tau_R \leq r_j$ and $\text{ORIG}(i) = \text{ORIG}(j)$: This arc represents the case where load i is the last load before a rest. After the rest, there is an empty move and load j.

Cost on all outgoing arcs from $p_{i,j}$ are -1. The shortest path from s to t gives the longest sequence of loads that can be covered by a driver. Note that because of the timing requirements the network is acyclic, ensuring the polynomial runtime of the shortest path algorithm.

The integer programming model and the Longest Path Heuristic can be extended to the case where there are more than two terminals. Arc conditions for two loads with destination of one equal to origin of the other will be the same. If there is need for empty moves in between, the travel time between the destination of one load to the origin of another will determine the feasibility and existence of arcs between the corresponding p nodes. The

number of nodes in the solution network depends only on k and the number of loads, not on the number terminals in the terminal network.

5.5 Special Case: 2DAP(V) with $k = 2$

In real-life systems, there are many lanes in the terminal network with drive time of about 5 hours, which allow the drivers to drive from one terminal to another and back. This especially applies to ABA laydown bid drivers. Therefore, analyzing the case where $k = 2$ is not only theoretically interesting, but has practical value as well.

5.5.1 Lower Bounds

It should be noted that any lower bound for 2DAP (*e.g.*, occupation bound, cycle bound, and improved cycle bound) is a valid lower bound for 2DAP(V) for any value of k. Moreover, 2DAP itself is lower bound of 2DAP(V) for all values of k. When $k = 2$, we can use the fact that the driver cannot move more than twice before going to rest to tighten the cycle bound.

Extended Cycle Bound

When a driver is dispatched, say from A to B, with load ℓ at time r_ℓ, that driver cannot be dispatched again from A to B until at least time $r_\ell + 2tt + \tau_R$, since the driver must travel from A to B with ℓ, and then from B to A with a load or empty, and then rest. If we count the total number of loads dispatched from A to B during the interval $[r_\ell, r_\ell + 2tt + \tau_R)$, we know we need unique drivers for each of them. Repeating this process for each load generates a lower bound. More formally, let

$$E_\ell = \{\ell' \in \mathcal{L} \mid orig(\ell') = orig(\ell), r_{\ell'} \in [r_\ell, r_\ell + 2tt + \tau_R)\} \tag{1}$$

and then a lower bound is

$$\underline{\mathrm{m}}^{EC} = \max_\ell |E_\ell| \tag{2}$$

5.5.2 Heuristics

Simple Heuristic

Simple Heuristic (SH) considers the loads originating from each terminal separately. If the origin of the loads are limited to only one terminal, say A, then we know that if we dispatch a driver d with a load ℓ, then the earliest time he can be dispatched, denoted by $\text{READY}(d)$, at A is $r_\ell + 2tt + \tau_R$, because the driver needs to come back with an empty move and go to rest when $k = 2$. For the driver's next assignment, we can focus on the earliest ready load after the ready time of the driver. Algorithm 5.4 provides the pseudo-code for the SH. $\mathcal{S}_d$ denotes the sequence of loads that the driver covers, m_s^T is the number of drivers used to cover loads originating from terminal $T = A, B$, and m_s the total number of drivers used.

Algorithm 5.4 Pseudo-Code of Simple Heuristic

1: $m_s^T \leftarrow 0$, $T = A, B$
2: **for** $(T = A, B)$ **do**
3: $\quad\mathcal{T} \leftarrow$ set of all loads originating at T
4: $\quad$**repeat**
5: $\quad\quad\text{READY}(d) \leftarrow 0$
6: $\quad\quad\mathcal{S}_d \leftarrow \emptyset$
7: $\quad\quad$**repeat**
8: $\quad\quad\quad$Let $\ell \in \mathcal{T}$ be the load with earliest ready time such that $r_\ell \geq \text{READY}(d)$
9: $\quad\quad\quad\mathcal{S}_d \leftarrow \mathcal{S}_d \cup \{\ell\}$
10: $\quad\quad\quad\text{READY}(d) \leftarrow r_\ell + 2tt + \tau_R$
11: $\quad\quad$**until** $\text{READY}(d) > \max_{i\in\mathcal{T}} r_i$
12: $\quad\quad\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{S}_d$
13: $\quad\quad m_s^T \leftarrow m_s^T + 1$
14: $\quad$**until** $(\mathcal{T} = \emptyset)$
15: **end for**
16: **return** $m_s = m_s^A + m_s^B$

Theorem 5.16. $m_s^T \leq \underline{m}^{EC}$, $T = A, B$.

Proof: Let $\mathcal{L}(T)$ denote the loads originating from terminal T, $T = A, B$. By construction of extended cycle bound, $|E_\ell| \leq \underline{\text{m}}^{EC}$, $\ell \in \mathcal{L}(T)$. We want to show that $|E_\ell \setminus \mathcal{S}_d| \leq \underline{\text{m}}^{EC} - 1$ for each $\ell \in \mathcal{L}(T) \setminus \mathcal{S}_d$, after a sequence $\mathcal{S}_d$ is constructed by Simple Heuristic (lines 7–11) from loads in $\mathcal{L}(T)$ and removed from $\mathcal{L}(T)$ (line 12).

Case 1: If $|E_\ell| < \underline{\text{m}}^{EC}$, then $|E_\ell \setminus \mathcal{S}_d| \leq \underline{\text{m}}^{EC} - 1$.

Case 2: If $\ell' \in \mathcal{S}_d \cap E_\ell$, then $|E_\ell \setminus \mathcal{S}_d| \leq |E_\ell| - 1 \leq \underline{\text{m}}^{EC} - 1$.

Case 3: If $|E_\ell| = \underline{\text{m}}^{EC}$, and $E_\ell \cap \mathcal{S}_d = \emptyset$, then there exists $\hat{\ell} \in \mathcal{S}_d$ such that $r_{\hat{\ell}} < r_\ell$. If such a load did not exist, then ℓ or a load with dispatch time r_ℓ would be selected as the

first load in $\mathcal{S}_d$ by line 8. Let $\hat{\ell} \in \mathcal{S}_d$ be the one with maximum dispatch time among these loads.

Since the construction step (lines 7–11) did not include any load in E_ℓ, it means that $r_{\hat{\ell}} < r_{\bar{\ell}} < r_{\hat{\ell}} + 2tt + \tau_R$, $\forall\, \bar{\ell} \in E_\ell$. Then, $\{\hat{\ell}\} \cup E_\ell \subset E_{\hat{\ell}}$, and $|E_{\hat{\ell}}| \geq |E_\ell| + 1 \geq \underline{\mathrm{m}}^{EC} + 1 > \underline{\mathrm{m}}^{EC}$, contradicting $|E_{\hat{\ell}}| \leq \underline{\mathrm{m}}^{EC}$.

The above argument shows that by constructing $\mathcal{S}_d$ and removing it form $\mathcal{L}(T)$ only once, we have $|E_\ell| \leq \underline{\mathrm{m}}^{EC} - 1$ for the all remaining loads ℓ. By applying the same argument to the remaining loads recursively, we can show that at each iteration the bound on $|E_\ell|$ goes down by at least 1. Therefore, after at most $\underline{\mathrm{m}}^{EC}$ iterations, $|E_\ell|$, $\ell \in \mathcal{L}(T)$ will be zero and all loads in $\mathcal{L}(T)$ will be assigned to a driver. Then, $m_s^T \leq \underline{\mathrm{m}}^{EC}$, $T = A, B$.

■

Theorem 5.17. *Simple Heuristic is a 2-approximation algorithm.*

Proof: Let m_{opt} denote the minimum number of drivers required to solve 2DAP(V) for $k = 2$. Simple heuristic uses $m_s = m_s^A + m_s^B$. By Theorem 5.16,

$$m_s = m_s^A + m_s^B \leq \underline{\mathrm{m}}^{EC} + \underline{\mathrm{m}}^{EC} \leq 2\, m_{opt}$$

■

It can be seen from Example 5.18 that the above bound is tight.

Example 5.18. 2 *loads.* $r_1 = 0$, $r_2 = tt$, $\textsc{orig}(1) = A$, $\textsc{orig}(2) = B$. *Then,* $m_{opt} = 1, m_s = 2$.

Observation 5.19. $m_{opt} \leq 2\, \underline{m}^{EC}$.

Proof: By Theorem 5.17, $m_s \leq 2\, \underline{\mathrm{m}}^{EC}$. Simple Heuristic produces an upper bound on the optimal (*i.e.*, $m_{opt} \leq m_s$). Then, $m_{opt} \leq 2\, \underline{\mathrm{m}}^{EC}$. ■

Example 5.5 provides a tight example to the above bound.

5.6 Computational Study for 2DAP and 2DAP(V)

In this section, we provide computational results that demonstrate the performance of lower bounds (occupation bound, improved cycle bound, and extended cycle bound), and heuristics (longest path heuristic, and simple heuristic).

The planning horizon is selected to be a week (*i.e.*, 168 hours). In each instance, dispatch times were generated based on a poisson process with number of loads 20, 30 and 40 from each terminal. $\tau_V = 11$ hours and $\tau_R = 10$ hours for instances of 2DAP(V), the current values set by U.S. DOT. Different instances were generated with travel time equal to 5, 3, 2.5, 2, and 1.1 hours, corresponding to $k = 2, 3, 4, 5$ and 10 respectively. Five instances were generated for each value of k and number of loads.

5.6.1 2DAP

We can apply the Longest Path Heuristic (Algorithm 5.3) to 2DAP by using a modified version of the model in Section 5.3.2 to find the longest sequence of loads. The nodes and arcs are the same as Section 5.3.2. Each outgoing arc from $p_j, j = 1 \ldots n$ has a cost of -1. The shortest path from s to t gives the longest sequence of loads that can be covered by a single driver. Note that the network is acyclic because of the timing requirements on the arcs and the shortest path can be found in polynomial time.

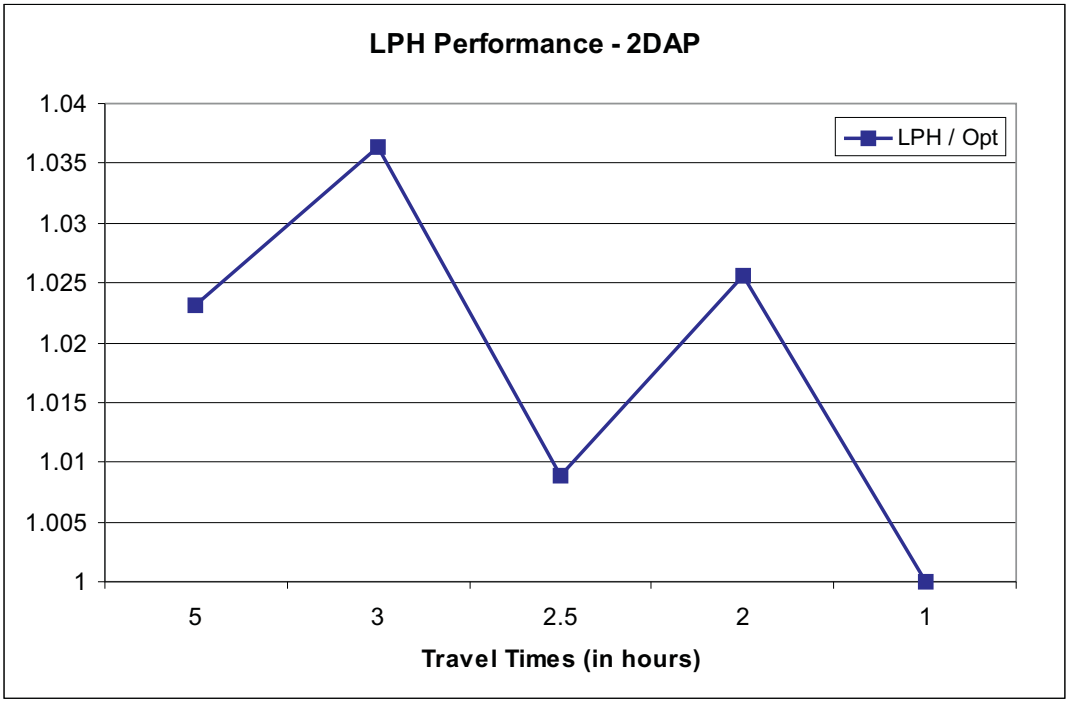

Figure 5.6: Average performance of Longest Path Heuristic for 2DAP.

As it can be seen from Figure 5.6 Longest Path Heuristic performs within 4% of the optimum on average for different travel times in 2DAP.

5.6.2 2DAP(V)

In this section, we analyze the performance lower bounds and heuristics related to 2DAP(V). Table 5.1 summarizes the results of the experiments. The columns shows the performance of the occupation bound, the extended cycle bound, improved cycle bound, longest path heuristic, and simple heuristic.

Table 5.1: Performance of lower bounds and heuristics for 2DAP(V).

k		Lower Bounds			Heuristics	
		$m_{opt}/\underline{\mathrm{m}}^{O}$	$m_{opt}/\underline{\mathrm{m}}^{EC}$	$m_{opt}/\underline{\mathrm{m}}^{IC}$	m_l/m_{opt}	m_s/m_{opt}
2	MAX	2.14	1.20	1.71	1.29	2.00
	MIN	1.13	1.00	1.00	1.00	1.44
	AVG	1.49	1.00	1.34	1.08	1.76
3	MAX	1.67	N/A	1.60	1.50	N/A
	MIN	1.00	N/A	1.00	1.00	N/A
	AVG	1.33	N/A	1.14	1.16	N/A
4	MAX	1.67	N/A	1.40	1.33	N/A
	MIN	1.00	N/A	1.00	1.00	N/A
	AVG	1.24	N/A	1.08	1.10	N/A
5	MAX	2.00	N/A	1.50	1.50	N/A
	MIN	1.00	N/A	1.00	1.00	N/A
	AVG	1.20	N/A	1.04	1.13	N/A
10	MAX	2.00	N/A	1.00	1.50	N/A
	MIN	1.00	N/A	1.00	1.00	N/A
	AVG	1.21	N/A	1.00	1.12	N/A

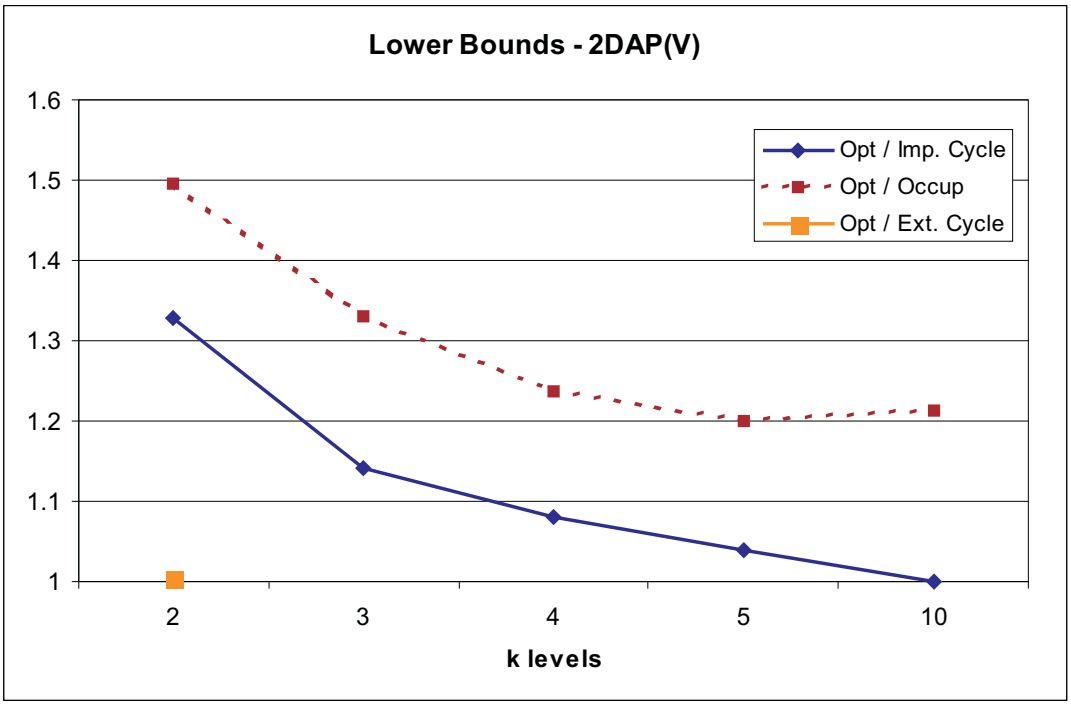

Figure 5.7: Average performance of lower bounds for 2DAP(V).

Figure 5.7 shows the average performance of the lower bounds. There is an apparent downward trend as k increases. As travel time decreases (*i.e.*, k increases), flexibility for driver schedules increases because there are more options for inserting rests within driver schedules. Therefore, fewer drivers may be required as travel time decreases. One interesting observation is that the extended cycle bound performs remarkably well for the $k = 2$.

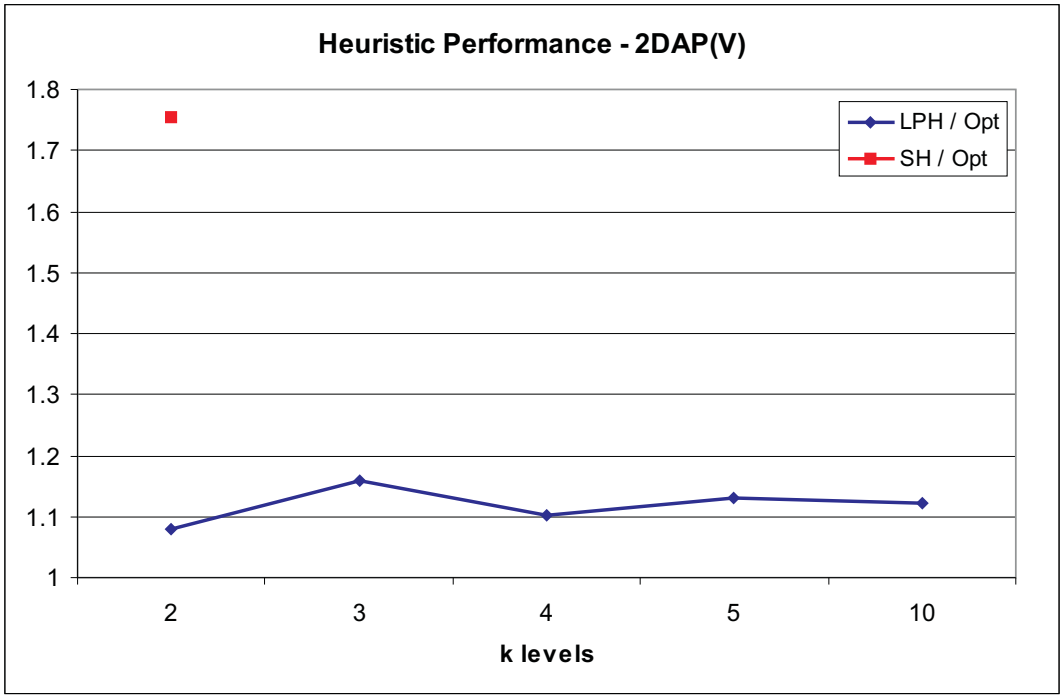

Figure 5.8: Average performance of Longest Path Heuristic and Simple Heuristic for 2DAP(V).

Figure 5.8 shows the average performance of Longest Path Heuristic and Simple Heuristic. LPH performs 16% of the optimal on average for all levels of k and as seen from Table 5.1, the worst performance observed for LPH is 50%.

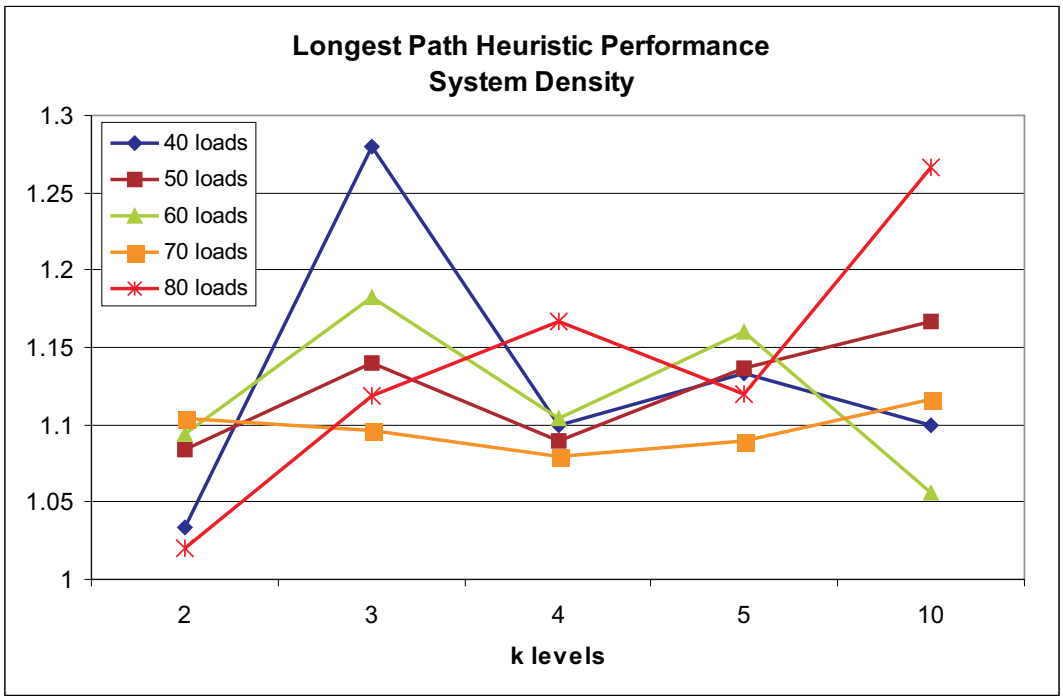

Figure 5.9: Average performance of Longest Path Heuristic for different system density levels

Figure 5.9 shows the average performance of the Longest Path Heuristic with respect to system density. As it can be seen from the figure, there is no apparent pattern in terms of performance related to the load density in the system. It is interesting to point out the best performance of LPH is achieved in the densest instance, the one with 80 loads and travel time 5 hours (*i.e.*, $k = 2$).

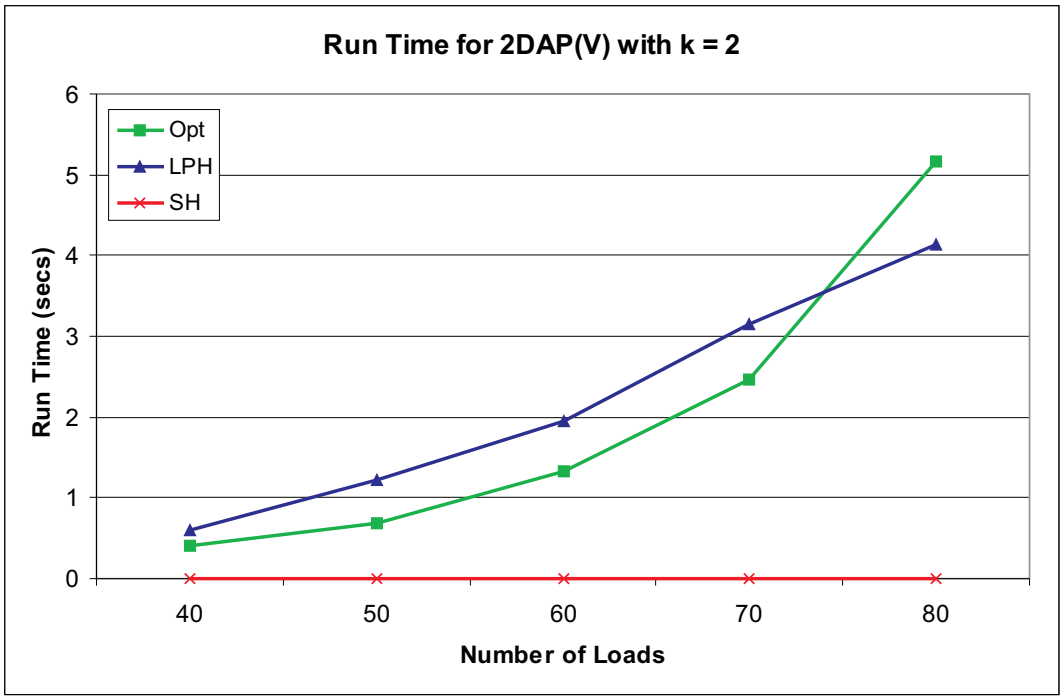

Figure 5.10: Run Time for 2DAP(V) with $k = 2$

Figures 5.10 and 5.11 show the running time of integer programming model, LPH and Simple Heuristic for the cases $k = 2$ and $k = 5$ respectively, with respect to the number of loads. As expected from an IP formulation, the running time shows an exponential increase as number of loads increases. LPH run time is almost linear and Simple Heuristic run time is negligible.

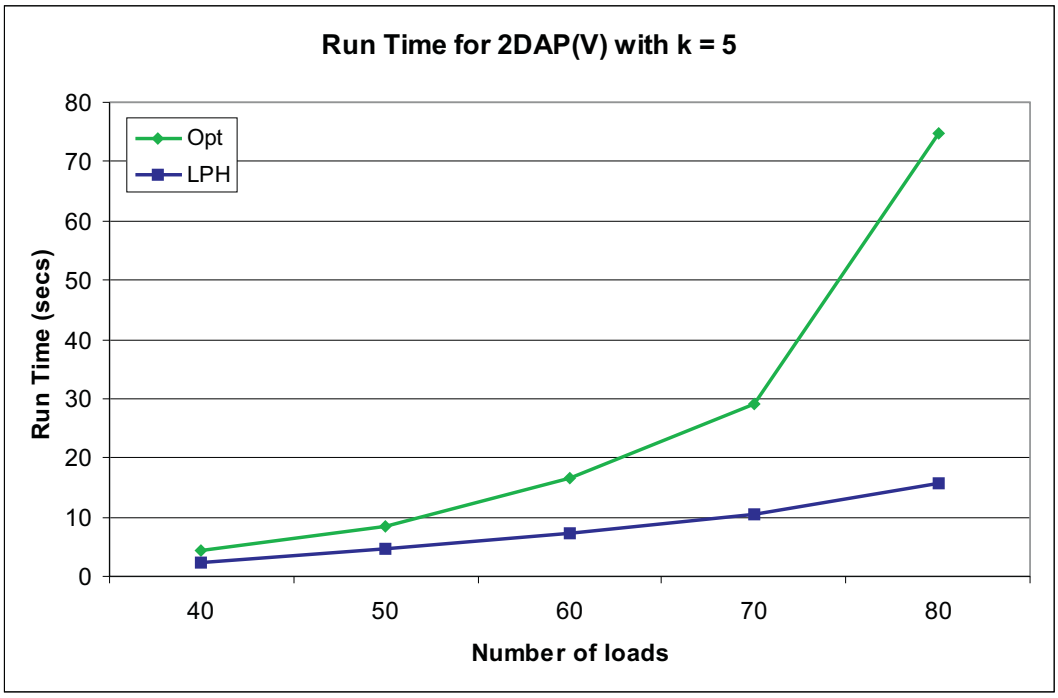

Figure 5.11: Run Time for 2DAP(V) with $k = 5$

There is noticeable increase in the running time from $k = 2$ to $k = 5$. This can be due to the increase in the number of nodes and arcs in the model (For two instances with same dispatch times and locations, the one with the shorter travel time tends to have more arcs, because shorter travel times implies more feasible couplings of loads). Even though this is also true for LPH, at each iteration of LPH, there is gradual decrease in the number of loads, and therefore in the nodes and arcs of the model.

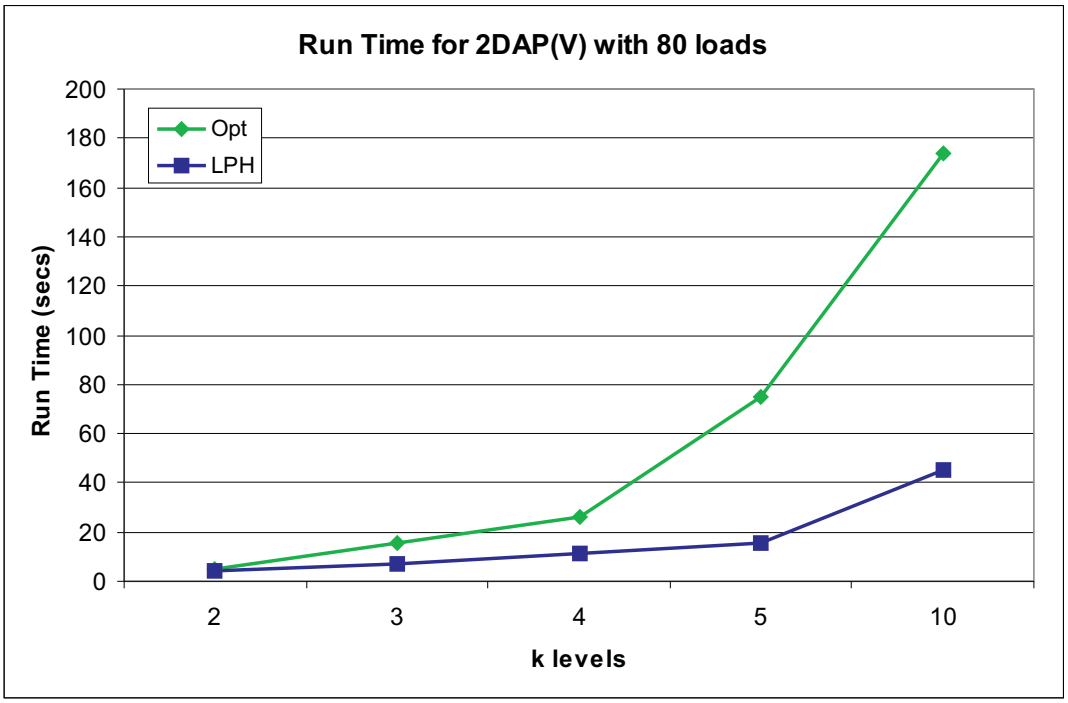

Figure 5.12: Run Time for 2DAP(V) with 80 loads

Figure 5.12 shows the running time of integer programming model and LPH with respect to k. We can observe the exponential increase in the running time of the IP model as k increases, which can again be explained by the increase in the number of nodes and arcs.

5.7 Solving 2DAP with Duty Time Restrictions (2DAP(U))

In this section, we consider 2DAP(U), the case where each driver cannot work more than τ_U hours before resting for a minimum of τ_R hours. In this case, the time a driver waits between moves counts towards his duty time.

Observation 5.20. *In 2DAP(U), the waiting times are important and need to be considered. If there is an empty move in between two loads, then regardless of the dispatch time of the empty move, the driver will incur the same waiting time. Therefore, we can assume that the empty move is executed immediately after the preceding load. If last move of a driver before rest is an empty move, then we can assume that it is immediately after is preceding*

load, resulting in no waiting time. Similarly, if the first move of a driver after rest is an empty move, we can assume that it is immediately before his second move, again resulting in no waiting time.

5.7.1 Integer Programming Formulation for 2DAP(U)

In this section, we propose an integer programming model, where all feasible sequences of loads that can be covered by a driver between two rests need to be generated a priori. Although problem structures are very similar between 2DAP(V) and 2DAP(U), we cannot use the model in Section 5.4.2. The reason is that the waiting times between loads accumulate in a driver's schedule. Therefore, determining if a load can be included in a driver's schedule not only depends on his current state and the last load he covered, but all the previous loads he has covered since his latest rest.

Therefore, we will propose a flow model, in which the nodes will correspond to all the possible "duties" for a driver. A duty is a sequence of load and empty moves that a driver can cover feasibly between two rests. Feasibility of a duty can be verified by checking the difference between the arrival time of the last load and the dispatch time of the first loads is less than τ_U.

Algorithm 5.5 and 5.6 give the pseudo-code for generating all feasible duties. S represents an ordered sequence of moves, with a load or empty. $\textsc{first}(S)$ and $\textsc{last}(S)$ indicate the first and last move in duty S respectively. $S + S'$ indicates that ordered sequence S' is appended to the end of S. $\textsc{record}(S)$ function records the duty S for future use in the flow model, and the pseudo-code for EXPAND function is given in Algorithm 5.6.

Algorithm 5.5 Duty Generation

```
for all (ℓ ∈ 𝓛) do
    S ← {ℓ}
    EXPAND (S)
end for
```

Let us assume that Algorithm 5.5 generated q duties, $S^j, j = 1 \dots q$. Let $\textsc{disp}(S)$ and $\textsc{arr}(S)$ denote the dispatch and the arrival time of duty S respectively (*i.e.*, $\textsc{disp}(S) = r_{\textsc{first}(S)}$ and $\textsc{arr}(S) = r_{\textsc{last}(S)} + tt$). We will construct a flow model using the generated

Algorithm 5.6 EXPAND function for 2DAP(U)

Require: $S,\ \tau_U$
  RECORD(S)
  $f \leftarrow$ FIRST(S)
  $e \leftarrow$ LAST(S)
  if $(r_e + tt - r_f \leq \tau_U - tt)$ **then**
    RECORD($\{$'empty'$\} + S$)
    RECORD($S + \{$'empty'$\}$)
  end if
  if $(r_e + tt - r_f \leq \tau_U - 2tt)$ **then**
    RECORD($\{$'empty'$\} + S + \{$'empty'$\}$)
  end if
  $t_{start} \leftarrow r_f,\ t_{end} \leftarrow r_e + tt$
  for all (ℓ such that $r_\ell \geq t_{end}$ and DEST(e) = ORIG(ℓ)) **do**
    if $(r_\ell + tt - t_{start} \leq \tau_U)$ **then**
      $S \leftarrow S + \{\ell\}$
      EXPAND (S)
    end if
  end for
  for all (ℓ such that $r_\ell \geq t_{end} + tt$ and ORIG(e) = ORIG(ℓ)) **do**
    if $(r_\ell + tt - t_{start} \leq \tau_U)$ **then**
      $S \leftarrow S + \{$'empty', $\ell\}$
      EXPAND (S)
    end if
  end for

duties. The model is similar to the one described in section 5.3.2. However, we have nodes for each duty instead of for each load.

Consider the solution of the minimum cost circulation on the following network $G = (\mathcal{N}, \mathcal{A})$ with nodes, s and t, and q nodes, $p^j, j = 1 \ldots q$, representing the duties $S^j, j = 1 \ldots q$.

The set of arcs includes an arc from s to each p_j with a cost of 1 and an upper bound of 1. There is an arc of cost 0 between p^i and p^j if ARR(S^i) + $\tau_R \leq$ DISP (S^j) and DEST($r_{\text{LAST}(S^i)}$) = ORIG($r_{\text{FIRST}(S^j)}$). Note that this timing requirement disables any arc between nodes corresponding to two duties that share a common load. All $p^j, j = 1 \ldots q$ are connected to t. Let x_{ij} denote the flow between p^i and p^j and y_i denote the flow from p^i to t. To ensure that every load is covered exactly once we need the following cover constraints:

$$\sum_{\{i|\ell \in S^i\}} \left(y_i + \sum_{j=1 \ldots q} x_{ij} \right) = 1,\ \forall \ell \in \mathcal{L}$$

With the addition of cover constraints, the optimal solution can have fractional flow. Therefore, we need to require the flow variables to be binary making the above formulation to an integer programming formulation.

Observation 5.21. *There is no dominating relation between 2DAP(V) and 2DAP(U), unless* $\tau_U = \tau_V$.

Example 5.22. *3 loads.* $r_1 = 0$, $r_2 = tt$, $r_3 = 2\,tt$,
$\text{ORIG}(1) = A$, $\text{ORIG}(2) = B$, $\text{ORIG}(3) = A$,
$\tau_V = 2\,tt$, $\tau_U = 3\,tt$.
In this instance, optimal solution to 2DAP(V) is two and optimal solution to 2DAP(U) is one.

Example 5.23. *3 loads.* $r_1 = 0$, $r_2 = tt$, $r_3 = 2\,tt$,
$\text{ORIG}(1) = A$, $\text{ORIG}(2) = B$, $\text{ORIG}(3) = A$,
$\tau_V = 3\,tt$, $\tau_U = 2\,tt$.
In this instance, optimal solution to 2DAP(V) is one and optimal solution to 2DAP(U) is two.

As it can be seen from Examples 5.22 and 5.23, we cannot claim that the solution of 2DAP(V) is a lower bound for 2DAP(U) or vice versa. If $\tau_V = \tau_U$, then any duty that is feasible for 2DAP(U) will also be feasible for 2DAP(V). However, in 2DAP(V), there can be driver schedules with too long waiting times that are not feasible for 2DAP(U). Therefore, solution of 2DAP(V) is a lower bound to 2DAP(U), if $\tau_V = \tau_U$.

5.7.2 Another Integer Programming Formulation for 2DAP(V)

The integer programming model in Section 5.7.1 can be used to solve 2DAP(V) with at most $k = \lfloor \tau_V / tt \rfloor$ moves between rests. Algorithm 5.7 gives the pseudo-code for the EXPAND function for 2DAP(V).

For a fixed k, the number of duties that will be generated by Algorithm 5.7 will be polynomial. In the worst case, the duty generation will generate all subsets of $\mathcal{L}$ with

Algorithm 5.7 EXPAND function for 2DAP(V)

Require: S, k
  RECORD (S)
  if $(|S| < k)$ **then**
    RECORD$(\{\text{'empty'}\} + S)$
    RECORD$(S + \{\text{'empty'}\})$
    $e \leftarrow$ LAST(S)
    $t_{end} \leftarrow r_e + tt$
    for all (ℓ such that $r_\ell \geq t_{end}$ and DEST$(e) =$ ORIG(ℓ)) **do**
      $S \leftarrow S + \{\ell\}$
      EXPAND (S)
    end for
    if $(|S| < k - 1)$ **then**
      RECORD$(\{\text{'empty'}\} + S + \{\text{'empty'}\})$
      for all (ℓ such that $r_\ell \geq t_{end} + tt$ and ORIG$(e) =$ ORIG(ℓ)) **do**
        $S \leftarrow S + \{\text{'empty'}, \ell\}$
        EXPAND (S)
      end for
    end if
  end if

cardinality less than or equal to k. Therefore, there will be at most

$$\sum_{i=1}^{k} \binom{n}{i} = c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0 = \mathcal{O}(n^k)$$

duties generated, for some constants c_0, $c_1 \ldots c_k$, which are not a function of k. For every duty generated, a similar one with an empty move appended to the beginning and end are also created. This does not effect the fact that number of duties generated for a fixed k is $\mathcal{O}(n^k)$.

5.7.3 Integer Programming Formulation for 2DAP(U,V)

With a minor modification to Algorithm 5.6, we can incorporate both drive and duty restrictions. The drive time restrictions require that a driver cannot have more that $k = \lfloor \tau_V / tt \rfloor$ moves between two rests. Therefore, we need to limit the number of moves within each duty. Algorithm 5.8 provides a pseudo-code for the modified EXPAND function for 2DAP(U,V).

Algorithm 5.8 EXPAND function for 2DAP(U,V)

Require: $S,\ \tau_U,\ k$
  RECORD (S)
  if $(|S| < k)$ **then**
    $f \leftarrow$ FIRST(S)
    $e \leftarrow$ LAST(S)
    if $(r_e + tt - r_f \leq \tau_U - tt)$ **then**
      RECORD$(\{\text{'empty'}\} + S)$
      RECORD$(S + \{\text{'empty'}\})$
    end if
    $t_{start} \leftarrow r_f,\ t_{end} \leftarrow r_e + tt$
    for all (ℓ such that $r_\ell \geq t_{end}$ and DEST$(e) =$ ORIG(ℓ)) **do**
      if $(r_\ell + tt - t_{start} \leq \tau_U)$ **then**
        $S \leftarrow S + \{\ell\}$
        EXPAND (S)
      end if
    end for
    if $(|S| < k - 1)$ **then**
      if $(r_e + tt - r_f \leq \tau_U - 2tt)$ **then**
        RECORD$(\{\text{'empty'}\} + S + \{\text{'empty'}\})$
      end if
      for all (ℓ such that $r_\ell \geq t_{end} + tt$ and ORIG$(e) =$ ORIG(ℓ)) **do**
        if $(r_\ell + tt - t_{start} \leq \tau_U)$ **then**
          $S \leftarrow S + \{\text{'empty'}, \ell\}$
          EXPAND (S)
        end if
      end for
    end if
  end if

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

This thesis describes a promising driver scheduling scheme for large-scale network dispatch problems based on combining greedy search ideas with partial solution enumeration. Results from computational study justify the algorithmic choices made, and a comparison with real-world dispatch data indicates that the quality of the solutions produced is very good. The heuristic can solve scheduling problems with 15,000–20,000 loads to be dispatched and thousands of drivers in approximately 5 minutes of PC computing time.

Although the scheme is designed to be implemented in a dynamic setting to provide real-time decision support, the technology can be used for tactical planning such as determining appropriate driver pool levels at various network terminals. Other uses include providing tactical driver planning support during periods of "unusual" operation, generating new driver plans when terminals change/close due to merger/divestiture activities, and generating new driver plans when U.S. DOT driver regulations change.

Several potential algorithmic extensions can be considered, including speeding up solution time through reuse of the enumeration trees and improving the solution quality by implementing the search in a greedy randomized adaptive search procedure (GRASP). Modeling pup-matching and load re-routing decisions within the driver scheduling scheme as an extension is also a direction for future research.

Driver fleet sizing and allocation for LTL carriers is a complex problem in large extent due to U.S. DOT regulations and union rules restricting the use of drivers. We have shown that tactical decisions related to the number of drivers required and where to domicile drivers in the linehaul network can be studied with driver scheduling technology that properly handles U.S. DOT regulations and union rules. Our study has demonstrated that union rules force LTL companies to employ substantially more drivers, and that poor driver allocation can significantly decrease the level of service.

Our technology can (and will be) used to analyze various other tactical decisions, such as the effect of changing primary terminals on lanes, varying the level of bid drivers in the system (which may help in union negotiations), and changing the set of driver domiciles.

Lastly, we focused on one of the fundamental problems of driver management: minimizing the number of drivers required to cover a given set of loads. We observed that the version with no restrictions on driver schedules is easy to solve. It remains an open question if the problem remains polynomially solvable when drive time restrictions or duty time restrictions are imposed on the driver schedules. Empirically, we observed that the problems are harder to solve with such restrictions. We presented intuitive polynomial-time heuristics that perform well in spite of this complexity, *e.g.*, the longest path heuristic.

REFERENCES

[1] Armacost, A. P., Barnhart, C., and Ware, K. A., "Composite variable formulations for express shipment service network design," *Transportation Science*, vol. 36, pp. 1–20, 2002. 6

[2] Barnhart, C., Cohn, A., Johnson, E. L., Klabjan, D., Nemhauser, G. L., and Vance, P., "Airline crew scheduling," in *Handbook of Transportation Science* (Hall, R. W., ed.), pp. 517–560, Norwell, MA: Kluwer Academic, 2003. 8

[3] Barnhart, C. and Schneur, R. R., "Air network design for express freight service," *Operations Research*, vol. 44, pp. 852–863, 1996. 6

[4] Çalışkan, C. and Hall, R. W., "A dynamic empty equipment and crew allocation model for long-haul networks," *Transportation Research Part A*, vol. 37, pp. 405–418, 2003. 7

[5] Corsi, T. M., "The truckload carrier industry segment," in *Trucking in the Age of Information* (Belman, D. and III, C. W., eds.), pp. 21–42, Burlington, VT: Ashgate, 2005. 1

[6] Crainic, T. G., "Service network design in freight transportation," *European Journal of Operations Research*, vol. 122, no. 2, pp. 272–288, 2000. 6

[7] Crainic, T. G., "Long-haul freight transportation," in *Handbook of Transportation Science* (Hall, R., ed.), pp. 451–516, Norwell, MA: Kluwer Academic, 2003. 6

[8] Crainic, T. G. and Roy, J., "Design of regular intercity driver routes for the LTL motor carrier industry," *Transportation Science*, vol. 26, no. 4, pp. 280–295, 1992. 7

[9] Dall'Orto, L. C., Crainic, T. G., Leal, J. E., and Powell, W. B., "The single node dynamic service scheduling and dispatching problem," *European Journal of Operations Research*, vol. 170, pp. 1–23, 2006. 6

[10] Elhallaoui, I., Villeneuve, D., Soumis, F., and Desaulniers, G., "Dynamic aggregation of set-partitioning constraints in column generation," *Operations Research*, vol. 53, pp. 632–645, 2005. 9

[11] Godfrey, G. and Powell, W. B., "An adaptive dynamic programming algorithm for single-period fleet management problems I: Single period travel times," *Transportation Science*, vol. 36, pp. 21–39, 2002. 9

[12] Godfrey, G. and Powell, W. B., "An adaptive dynamic programming algorithm for single-period fleet management problems II: Multiperiod travel times," *Transportation Science*, vol. 36, pp. 40–54, 2002. 9

[13] Papadaki, K. and Powell, W. B., "An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem," *Naval Research Logistics*, vol. 50, pp. 742–769, 2003. 6

[14] Powell, W. B., "A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers," *Transportation Science*, vol. 30, no. 3, pp. 195–219, 1996. 7

[15] Powell, W. B., "Dynamic models of transportation operations," in *Handbooks in Operations Research and Management Science: Supply Chain Management* (Graves, S. and de Tok, T. A. G., eds.), pp. 677–756, Amsterdam: Elsevier, 2003. 9

[16] Powell, W. B., Shapiro, J. A., and Simao, H. P., "An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem," *Transportation Science*, vol. 36, pp. 231–249, 2002. 9

[17] Powell, W. B. and Topaloglu, H., "Stochastic programming in transportation and logistics," in *Stochastic Programming* (Shapiro, A. and Ruszczynski, A., eds.), Handbooks in Operations Research and the Management Sciences, pp. 555–635, Amsterdam: Elsevier Science, 2003. 9

[18] Schaefer, A. J., Johnson, E. L., Kleywegt, A. J., and Nemhauser, G. L., "Airline crew scheduling under uncertainty," *Transportation Science*, vol. 39, pp. 340–348, 2005. 8

[19] Yang, J., Jaillet, P., and Mahmassani, H. S., "Real-time multivehicle truckload pickup and delivery problems," *Transportation Science*, vol. 38, pp. 135–148, 2004. 9